Inferring and Predicting Dynamic Representations for Structured Temporal Data

Edouard Delasalles

THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ

Spécialité
Informatique
École Doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Edouard Delasalles

Pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Sujet de la thèse :

Inferring and Predicting Dynamic Representations for Structured Temporal Data

Le jury sera composé de :

M. Alexandre Allauzen	ESPCI - CNRS	Rapporteur
M. Thierry Artieres	Aix Marseille Université - ECM - LIS	Rapporteur
Mme Ahlame Douzal	Université Grenoble Alpes - LIG	Examinatrice
M. Patrcik Gallinari	Sorbonne Université - LIP6	Examinateur
M. Ludovic Denoyer	Sorbonne Université - LIP6	Directeur de thèse
M. Sylvain Lamprier	Sorbonne Université - LIP6	Encadrant de thèse

ABSTRACT

Temporal and sequential data constitute a large part of data collected digitally. Predicting future values of such data is an important and challenging task in domains such as climatology, optimal control, or natural language processing. Standard statistical methods are based on linear models and are often limited to low dimensional data. We instead use deep learning methods that are more capable of handling structured high dimensional data and leverage large quantities of training examples.

In this thesis, we are interested in latent variable models. Contrary to autoregressive models that directly use past data to perform prediction, latent models infer low dimensional vectorial representations of data on which prediction or imputation are performed. Latent vectorial spaces allow us to learn simple dynamic models that are then able to generate high-dimensional and structured data.

In the first part, we propose a structured latent model for spatio-temporal data forecasting. Given a set of spatial locations where data such as weather or traffic are collected, we infer latent variables for each location and use spatial structure in the dynamic functions. The model is also able to discover correlations between series without prior spatial information.

In the second part, we focus on predicting data distributions, rather than point estimates as done in the first part. To do so, we propose a latent model that generates latent variables used to condition a generative model. We use text data to evaluate our model on the task of diachronic language modeling.

In the last part, we propose a stochastic prediction model. This is a latent model that uses the first values of sequences to generate several possible futures. Here, the generative model is not conditioned to an epoch, like is the second part, but to new sequences. We apply this model to the challenging task of stochastic video prediction.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AI	Artificial Intelligence
AR	Autoregressive
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
BiRNN	Bidirectional RNN
BPTT	Backpropagation Through Time
BoW	Bag of Words
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DTW	Dynamic Time Wrapping
ELBO	Evidence Lower Bound
EM	Expectation Maximization
FVD	Fréchet Video Distance
GAN	Generative Adversarial Model
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
KLD	Kullback-Leibler Divergence
KNN	K-Nearest Neighbours
LDA	Latent Dirichlet Allocation
LM	Language Model
LPIPS	Learned Perceptual Image Patch Similarity
LSTM	Long Short-Term Memory
MA	Moving Average
ML	Machine Learning
MF	Matrix Factorization
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
NAG	Nesterov's Accelerated Gradient

NLP	Natural Language Processing
NN	Neural Network
NYT	New York Times
ODE	Ordinary Differential Equation
PCA	Principal Component Analysis
PSNR	Peak Signal-to-Noise Ratio
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RL	Reinforcement Learning
RMSE	Rooted Mean Square Error
RNN	Recurrent Neural Network
S2	Semantic Scholar
seq2seq	sequence to sequence
SGD	Stochastic Gradient Descent
SSIM	Structured Similarity
SSM	State Space Model
SVM	Support Vector Machine
VAE	Variational Auto-Encoder
VI	Variational Inference

CHAPTER 1

INTRODUCTION

Contents

1.1 Context

Artificial Intelligence (AI) aims at constructing autonomous systems that are capable of reproducing human cognitive functions to solve high-level tasks. This concept was born together with the introduction of computers several decades ago and is gaining a rapidly growing interest. AI is a vast domain in computer science, encompassing many sub-fields such as, but not limited to, knowledge representation, reasoning, natural language processing, robotics, or multi-agent systems.

Among them, Machine Learning (ML) and Deep Learning (DL) have gained a rapidly growing interest: for instance, the number of participants at the NeurIPS conference went from around one thousand in 2010 to over six thousand in 2019. ML and DL methods leverage the overwhelming quantity of digital information produced by human activity and natural phenomena monitoring, to learn decision and regression models.

A large part of the tremendous quantity of digital data has a temporal component. In this thesis, we propose methods to leverage it to build autonomous systems and to improve our comprehension of environmental, societal, and human dynamics.

1.1.1 Temporal Data

Nearly all digital data samples have some form of temporal annotation. For instance, digital photos or computer logs are marked with timestamps. And nearly all textual documents come with at least some idea of their writing time, and often with a precise year for documents from the last centuries. The temporal component can also come from the semantics of data. People's age or the production year of cars are other examples of temporal information that can be leveraged to improve models' performances.

There are also classical time series, present in many domains such as ecology, meteorology, biology, medicine, economics, traffic, and vision, collected by an increasing number of sensors disseminated around and above the world. As an example, NASA has currently more than 27 earth observation satellites in orbit [1]. Moreover, there are weather stations monitoring temperature, wind directions, or atmospheric pressure. And with the democratization of video cameras, there are 500 hours of videos uploaded on youtube every minute in 2019 [2].

1.1.2 Temporal Tasks

Most ML models rely on a vectorial representation of data (feature vectors) to learn decision boundaries or regressors in feature spaces. The classic method to extract a vector representation from a time series is by taking its frequency spectrum through Fourier Transform. However, this method is not adapted for online tasks such as prediction or filtering, as the sequential structure of time series is lost.

Time series are often of varying size and sometimes sampled at different temporal frequencies. And different temporal acquisition modalities gave rise to different task categories and a vast spectrum of methods to solve them. We now briefly present three categories of such temporal tasks: smoothing, filtering, and prediction.

In smoothing tasks, complete sequences are available and are processed as a whole. It encompasses tasks such as anomaly detection, time series modeling, or data imputation. Classical ML methods include Bag of Words (BoW) and Dynamic Time Wrapping (DTW) (Yazdi et al. 2018) that extract representations used as inputs of classical ML models (K-Nearest Neighbours (KNN), Support Vector Machine (SVM), Random Forests).

Filtering tasks correspond to online acquisition protocols. Data arrive in a streaming fashion, one timestep at a time, and the goal is to infer some hidden attributes of the new sample. This setting corresponds to tasks such as tracking,

1. https://en.wikipedia.org/wiki/List_of_Earth_observation_satellites
2. https://www.youtube.com/about/press/

control, or navigation. Classical statistical models used to solve these tasks are particle filters, Kalman filters, or Hidden Markov Model (HMM).

Lastly, prediction tasks consist of predicting future values given past ones. Applications are in weather forecasting, stock market prediction, or video prediction. Generally, a prediction task can be formulated from any temporal dataset. Classical methods are autoregressive methods that learn linear mappings between past and future samples.

In this thesis, we propose to explore DL and Representation Learning methods to learn generic representations of temporal data, together with dynamic models in a self-supervised manner. These representations can then be used for the different problems presented previously, and offer a general way to approach temporal tasks.

1.1.3 Deep Learning and Temporal Data

In the past decade, the DL paradigm emerged and showed previously unattained performances on many ML tasks. While traditional ML algorithms are usually trained on hand-crafted features, DL methods can learn meaningful representations directly from raw signals such as pixels, audio waves, or text. This representation learning technique is now at the heart of numerous works, notably image classification, language modeling, or reinforcement learning.

The first breakthrough in DL comes from the computer vision community and the introduction of the Convolutional Neural Network (CNN) (LeCun et al. 1989). Leveraging millions of annotated images, and the computational power of Graphical Processing Units (GPUs), Krizhevsky et al. 2012 won the 2012 ImageNet classification challenge by a large margin. Following this achievement, researchers are exploring the incorporation of deep representation learning in nearly all ML tasks.

For temporal data, most DL models are built around Recurrent Neural Networks (RNNs). They are flexible models that process temporal samples sequentially and produce latent representation at each timestep. They can thus model sequences of varying sizes and can be easily formalized for smoothing, filtering, and prediction tasks. Moreover, their recurrent architecture allows them to model complex conditional probability distributions. They are used for several sequential tasks such as language modeling, speech recognition, or video prediction for instance. We will detail some of them in Chapter 2.

RNNs are powerful and versatile but are not suited for every temporal task. One of their main drawbacks is their tendency to overfit the training data, making temporal extrapolation difficult. Another is that their architecture does not allow them to handle data structure natively, such as the spatial structure of spatio-temporal data.

1.2 Contributions

1.2.1 Designing Spatio-Temporal Neural Networks

In Chapter 3, we study time series that exhibit spatial dependencies. This kind of series is present in many domains such as meteorology, or traffic. Modeling this kind of series raises several challenges. They are multivariate time series, with usually a large number of series that present complex temporal dependencies schemes. Moreover, these series often exhibit complex dynamics and are often subject to noises. Answering these challenges leads to consider DL methods.

For this contribution, we introduced a general class of deep spatio-temporal models for time series of spatial processes. They allow us to explicitly model both spatial and temporal dependencies. We focus in this chapter on two tasks: forecasting (prediction) and missing data imputation (smoothing). We propose a model able to capture the dynamics and correlations in multiple series at the spatial and temporal levels. Besides reporting a significant improvement over traditional ML approaches, and more recent DL ones, we also show that the model is able to discover relevant spatial relations between series.

This line of research led to a conference paper:

> Ali Ziat*, Edouard Delasalles*, Ludovic Denoyer, and Patrick Gallinari (2017). "Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery". In: *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pp. 705–714,

and a journal extension:

> Edouard Delasalles, Ali Ziat, Ludovic Denoyer, and Patrick Gallinari (2019c). "Spatio-temporal neural networks for space-time data modeling and relation discovery". In: *Knowledge and Information Systems* 61.3, pp. 1241–1267.

1.2.2 Learning Dynamic Representation of Structured Data Distributions

In the previous contribution, we learned deterministic representations and dynamics. For each timestamp, a single point estimate of the future value of a spatio-temporal time series is produced. However, real-world data often follow stochastic generation processes. Hence, estimating the distribution of data points at each timestep, and being able to predict the distribution's evolution, is a key-challenge for modeling data through time.

In Chapter 4, we propose to combine temporal representations learning and generative probabilistic models. We applied our model to textual data. Language

is affected over time by various shifts; the meaning of words can shift, new words appear as other vanish, and yesterday topics are different from tomorrow's ones (Aitchison 2005).

This work led to a first publication, where we use global variables to model the temporal dynamics of language:

> Edouard Delasalles, Sylvain Lamprier, and Ludovic Denoyer (2019a). "Dynamic Neural Language Models". In: *Neural Information Processing - 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12-15, 2019, Proceedings, Part III*, pp. 282–294.

In a second publication, we also took into account the authors of documents. It allowed us to learn dynamic representations of authors:

> Edouard Delasalles, Sylvain Lamprier, and Ludovic Denoyer (2019b). "Learning Dynamic Author Representations with Temporal Language Models". In: *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pp. 120–129.

1.2.3 Stochastic Prediction

In the two first contributions, we were interested in modeling temporal data at a large temporal scale, mainly at the year level. This led us to consider time as a global attribute shared by all data samples. However, temporal phenomena also occur at inferior time scales.

In our third contribution, presented in Chapter 5, we propose to model the dynamics of local temporal events. To study this kind of event, we tackled the problem of stochastic prediction of videos. In a video dataset, each sample lives in its own temporal referential. However, all videos share similarities. For instance, in a bouncing ball dataset, the gravity always affects the ball the same way. These kinds of features common to all samples can be leverage by ML algorithms to learn prediction models.

An interesting feature of natural videos is their inherent stochasticity. Even if two samples share similar starting frames, the subsequent ones might be very different. It is thus necessary and challenging for a prediction model to handle this temporal stochasticity. More generally, it challenges the ability of a model to capture visual and dynamic representations of the world.

While most state-of-the-art approaches are based on autoregressive models built around RNNs, we propose a novel stochastic dynamic model that performs prediction in a low-dimensional latent space. It is a residual dynamic model that takes inspiration from recent advances relating to residual networks (K. He et al. 2016) and Ordinary Differential Equations (ODEs) (Chen et al. 2018).

This work led to a conference paper:

Jean-Yves Franceschi*, Edouard Delasalles*, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (2020). "Stochastic Latent Residual Video Prediction". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*.

CHAPTER 2

RELATED WORK

Contents

In this thesis, we are interested in modeling data that evolve through time. As described in the introduction, temporal data are involved in many different tasks: sequence classification, sequence matching, anomaly detection, reinforcement learning, etc... In this work, we focused on self-supervised tasks, where target data are portions of source data, and not an external human-produced label that is costly to acquire. As we will see, self-supervised tasks, like prediction, can have direct applications, but they are also useful for downstream tasks. Indeed, they are often used to extract high-level features than can then be used for downstream classification tasks for instance.

In the first section of this related work, we will introduce our tasks and notations, together with statistical and machine learning models for time series.

Secondly, we will present Recurrent Neural Networks (RNNs) with their different architectures and applications. Lastly, we will present deep variational approaches for sequential data.

2.1 Time Series

Time series are present in many fields, for instance in medical-biology with electroencephalograms or electrocardiograms, or climate with sea surface temperature or wind speed for instance. Formally, we define a time series $\mathbf{x}$ as an ordered set of real-valued vectors $\mathbf{x}_t \in \mathbb{R}^n$: $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$. In this thesis, we will only consider time series sampled uniformly in time, such that the time elapsed between two samples $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ is always the same. Note that this is not always the case. For instance, information diffusion processes often occur in continuous time and happen in an asynchronous fashion (Saito et al. 2009).

There are many tasks associated with time series: classification, prediction, imputation, indexation, segmentation, anomaly detection, etc... In this thesis, we focus on two tasks: prediction and imputation. We describe two families of methods to solve these tasks: autoregressive methods and state-space models.

2.1.1 Temporal Tasks

Prediction

Prediction tasks consist of predicting future values of sequences. The task in itself as a lot of direct applications for instance in stock market prediction, or weather forecasting, but also for other temporal tasks. For instance, model-based reinforcement learning models have to predict future states of the world in order to choose the best possible action.

In statistics, classical linear models are based on autoregressive and moving average components, described in Section 2.1.2. Most assume linear and stationary time dependencies with a noise component (Gooijer et al. 2006). In Machine Learning (ML), non-linear extensions of these models based on Neural Networks (NNs) were proposed as early as the nineties, opening the way to many other non-linear models developed both in statistics and ML, like kernel methods (Müller et al. 1999) for instance. In this thesis, we will investigate Deep Learning (DL) approaches, based on RNN and dynamic latent models, described in the following sections of this related work.

Imputation

In imputation tasks, we do not have access to data points at every timestep. The goal is to impute the values of those missing data. This task covers a wide area of problems and situations.

A category of data particularly affected by missing value is spatio-temporal data. This type of data is often acquired by networks of physical sensors. These sensors are not always reliable, they can stop recording or transmitting data, or data can be too noisy to provide useful information. Observations can also be blurred or occulted by external factors. For instance, satellite imaging in the visible domain is sensible to clouds that occult parts of the earth's surface. When needed, this information should be reconstructed using available data recorded at different times and locations, or data coming from other types of sensors. Another example that often occurs in traffic applications is when no signal is recorded at some places because of the absence of vehicles equipped with sensors at these places. This does not mean, of course, that traffic is absent. Hence, the values should be inferred from data available at other places.

The structure of time series allows for many simple heuristics to work well for the imputation task. The most simple one is the mean of previous and next values, which assumes linear dependencies between values. Another strategy is to fill the missing values with the last observed values, which assumes stationary time series. In the case of multivariate time series, another strategy is to perform knn-substitution, which consists of replacing missing values of a series by one of the closest series, for a given metric.

Classical ML algorithms for missing values can be also applied for time series imputation. The canonical approaches are based on the Expectation Maximization (EM) algorithm and Matrix Factorization (MF) methods. Bańbura et al. 2014 proposed an adaptation of the EM algorithm for time series with missing data, claiming good results for long consecutive missing values. And recently, several adaptations of MF have been proposed for data completion in time series (Y. Song et al. 2012; Shang et al. 2014; W. Shi et al. 2016). In this thesis, we will mostly compare our work to RNN based methods, described in Section 2.2

2.1.2 Autoregressive Models

Autoregressive models learn a prediction function f that takes as inputs past values of the series and predict the next one: $\mathbf{x}_{t+1} = f(\mathbf{x}_{t-k}, \ldots, \mathbf{x}_t)$. Here, k is called the order of the prediction model, which is the number of previous timesteps on which depends the prediction. To learn this function, we have access to a historic $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, on which we fit the function f. The challenge of this task is to learn a prediction function that generalizes to future unseen values $\mathbf{x}_{T+1}, \mathbf{x}_{T+2}, \ldots$.

Statistical forecasting models rely on strong assumptions to achieve this objective. The standard Autoregressive (AR) model assumes linear autocorrelation of time series plus a stochastic process. The AR model of order p writes as follows:

$$\mathbf{x}_t = \sum_{k=1}^{p} \boldsymbol{\theta}_k \mathbf{x}_{t-k} + \boldsymbol{b} + \epsilon_t. \tag{2.1}$$

Here, $\boldsymbol{\theta}_k \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$ are the parameters of the model, and ϵ_t is white noise.

The learning problem associated with this model can be formulated as a least-squares regression problem as follows:

$$\Theta^*, \boldsymbol{b}^* = \underset{\Theta=\{\theta_1,\dots,\theta_p\},\boldsymbol{b}}{\arg\min} \sum_{t=p}^{T-1} ||\mathbf{x}_{t+1} - \sum_{k=1}^{p} \boldsymbol{\theta}_k \mathbf{x}_{t-k} - \boldsymbol{b}||_2^2.$$

Since this is a classic convex least-squares regression problem, the parameters can found analytically or learned by Stochastic Gradient Descent (SGD). Afterward, prediction can be performed by recursively applying Equation 2.1. Since the model is differentiable given its inputs, it is possible to parameterize it with more complex functions, like NNs for instance. In this case, the objective function becomes non-convex, and SGD is required to estimate the parameters.

The classical AR process depends linearly only on its past values, which prevents it from modeling abrupt changes in the process. Indeed, a one-time large disruption in the series behavior can affect the model infinitely far into the future. To model these shocks more accurately, AR models are often used together with Moving Average (MA) models to form the Autoregressive Moving Average (ARMA) model.

In a MA model, the next value depends on the series mean, and on a linear transformation of the last q error terms:

$$\mathbf{x}_t = \mu + \sum_{k=1}^{q} \phi_k \epsilon_{t-k} + \epsilon_t$$

where ϕ_k are the parameters, ϵ_t are white noise error terms, and μ is the mean of the series. In this model, since the next value depends directly on the past innovations ϵ_{t-k}, a brutal shock only affects the model for the next q timesteps.

The ARMA model is the combination of two previously described models:

$$\mathbf{x}_t = \sum_{k=1}^{p} \boldsymbol{\theta}_k \mathbf{x}_{t-k} + \sum_{k=1}^{q} \phi_k \epsilon_{t-k} + \epsilon_t + \boldsymbol{b}.$$

It benefits from both the AR and MA properties, and the parameters can be estimated by maximum likelihood.

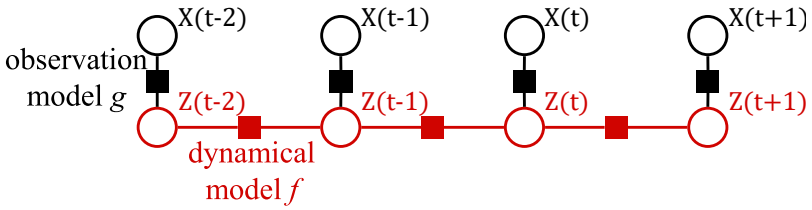


Figure 2.1 – The dynamic factor graph model from Mirowski et al. 2009. Latent states are inferred together with a dynamic latent model by energy minimization with gradient descent. Illustration taken from Mirowski et al. 2009

The ARMA model has several extensions in the literature, with notably the Autoregressive Integrated Moving Average (ARIMA) model. The model acts on differenced series, instead of the values themselves. Differencing consists in computing the differences between consecutive values of a time series. It means that instead of considering the raw values $\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots$, the model takes as inputs the differences $\mathbf{x}_t - \mathbf{x}_{t-k}, \mathbf{x}_{t+1} - \mathbf{x}_{t-k+1}, \mathbf{x}_{t+2} - \mathbf{x}_{t-k+2}, \ldots$. Differencing non-stationary time series several times may yield a stationary representation of it, thus enabling the use of ARMA. This is the case for seasonal time series for instance. In this case, by applying differentiation with order k equal to the time series seasonality length removes the seasonality, and the process becomes stationary.

2.1.3 Latent Variable Models

In autoregressive approaches, each predicted data point $\mathbf{x}_{t+1}$ has to be fed back to the model in order to produce the next prediction $\mathbf{x}_{t+2}$. Hence, their performance for long term prediction is tightly bounded to their capacity to generate realistic data. Errors can accumulate quickly, and lead to computational instabilities.

A different approach consists in learning a dynamic function in a latent space. Such models decouple dynamics from the data generation process. The objective of latent dynamic models is to infer a latent vectorial representation $\mathbf{z}_t$ of data points $\mathbf{x}_t$, that follow a dynamic model $f : \mathbb{R}^l \rightarrow \mathbb{R}^l$ such that $\mathbf{z}_{t+1} = f(\mathbf{z}_t)$. Classical latent variables ML are State Space Models (SSMs) and Hidden Markov Models (HMMs). In this thesis, we are interested in DL approaches, like the Dynamic Factor Graph model from Mirowski et al. 2009.

They proposed a dynamic factor graph model, that learn to infer latent temporal variables by energy minimization with gradient descent. A schematic view can be seen in Figure 2.1. The energy to minimize has the form:

$$E(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{z}) = \sum_{t=1}^{T} ||g_{\boldsymbol{\phi}}(\mathbf{z}_t) - \mathbf{x}_t||_2^2 + ||f_{\boldsymbol{\theta}}(\mathbf{z}_{t-1}) - \mathbf{z}_t||_2^2.$$

The latent states $\mathbf{z}_t$ are hence constraint to be decoded into observations $\mathbf{x}_t$ with g_ϕ, and also to be predictable by the dynamic model f_θ. They also introduce a smoothness penalty on latent variables, in order the regularize both the latent states and the dynamic function:

$$R(\mathbf{z}) = \sum_{t=1}^{T} = \sum_{t=1}^{T} ||\mathbf{z}_{t-1} - \mathbf{z}_t||_2^2.$$

The algorithm was successfully applied to synthetic and real-world datasets, notably to motion capture datasets.

This idea was extended in Ziat et al. 2016 for spatio-temporal time series. Instead of learning a unique latent vector per timestep, they propose to learn different latent vectors for different spatial locations. They have access to a binary adjacency matrix $\boldsymbol{A}$ where $\boldsymbol{A}_{i,j} = 1$ means that series are "close" in space. The closeness is usually defined by a hand-tuned distance threshold between locations. With this added spatial information, they propose to optimize the following loss:

$$E(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{z}) = \sum_{i=1}^{N}\sum_{t=1}^{T} ||g_\phi(\mathbf{z}_t^i) - \mathbf{x}_t^i||_2^2 + \sum_{i=1}^{N}\sum_{t=1}^{T} ||f_{\boldsymbol{\theta}}(\mathbf{z}_{t-1}^i) - \mathbf{z}_t^i||_2^2 + \sum_{i=1}^{N}\sum_{j=1}^{1}\sum_{t=1}^{T} \boldsymbol{A}_{i,j}||\mathbf{z}_t^i - \mathbf{z}_t^j||_2^2.$$

The idea is to regularize the latent space by keeping series closed in the observed space also closed in the latent space. They successfully applied the model road traffic prediction and imputation.

2.2 Recurrent Neural Networks

Sequence modeling tasks have gained a lot of interest in the DL community, and most of the works rely on some variation of Recurrent Neural Networks. For instance, Che et al. 2016 use them for data imputation in health-care related tasks, (X. Shi et al. 2015) for spatio-temporal modeling, or (Srivastava et al. 2015) for video prediction. In this section, we present in detail the RNN and its main implementation: the Long Short-Term Memory. We also present in more detail the application of RNNs to language modeling. Langue modeling is one of the tasks where RNNs show the best performances and is the application of one of the chapters of this thesis. Moreover, it allows us to present some regularization technics specific to RNNs.

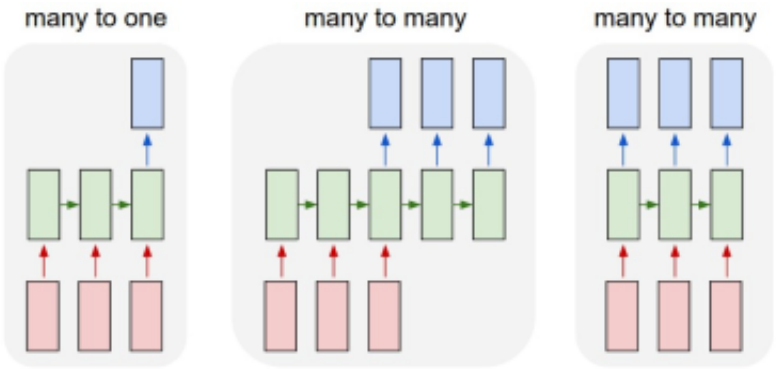


Figure 2.2 – Diagram of RNN. Their flexible design enables them to be used in diverse configurations, allowing them to be used in many tasks where sequences are involved. Illustration taken from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

2.2.1 Models and Architectures

Recurrent Neural Networks are a class of artificial neural networks designed to handle sequential data. They process each element of a sequence one after the other and maintain a hidden vectorial representation of the sequence at each timestep. These internal representations act as memories of the previously seen inputs and can be used as input for downstream tasks.

RNNs have been used for time series modeling in different contexts since the early nineties (Connor et al. 1994). Recently, these models have witnessed important successes for several sequence modeling problems; their flexible design allows them to model a large range of data, and solve various sequential tasks. For instance, in Figure 2.2, we can see different configurations of RNNs for different tasks. This led to breakthroughs in domains like speech recognition (Graves et al. 2005; Graves et al. 2013), Natural Language Processing (NLP) (Mikolov et al. 2010; Mikolov et al. 2012; Sutskever et al. 2011; Cho et al. 2014), video prediction (X. Shi et al. 2015; Y. Wang et al. 2017), and many others. They are also used as building blocks in many applications, such as reinforcement learning (Wierstra et al. 2010), sequential image generation (Gregor et al. 2015), or image captioning (Vinyals et al. 2017).

We now give a formalization of the RNN. Classical tasks solved by RNNs often involve the prediction of a target variable $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$ with $\mathbf{y}_t \in \mathbb{R}^m$. The goal of an RNN is to maximize the log-likelihood of targets factorized as follows:

$$\log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) = \sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{x}_{1:t}), \tag{2.2}$$

where $\mathbf{x}_{1:t} = \mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_t)$ are the inputs of the model.

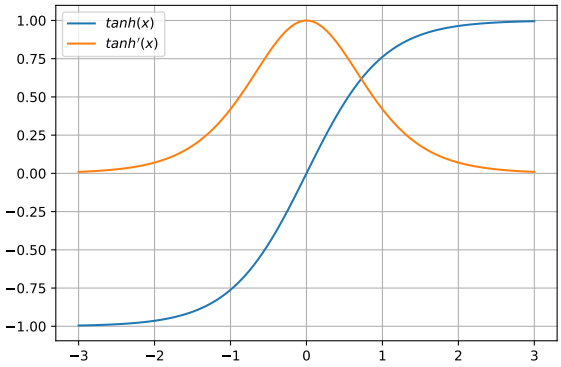


Figure 2.3 – The *tanh* function and its derivative on the interval $[-3, 3]$. We can see that the derivative is close to 0 on both ends. This behavior is referred as activation's saturation that prevents gradient from flowing through it.

To model the conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{x}_{1:t})$, the RNN maintains a hidden state $\boldsymbol{h}_t \in \mathbb{R}^d$ such that $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{x}_{1:t}) = p_{\boldsymbol{\theta}}(\mathbf{y}_t|\boldsymbol{h}_t)$. This hidden state is updated with each new $\mathbf{x}_{t+1}$ with a function $f_{\boldsymbol{\theta}}$ such that $\boldsymbol{h}_t = f_{\boldsymbol{\theta}}(\mathbf{x}_t, \boldsymbol{h}_{t-1})$ and $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\boldsymbol{h}_t)$ is parameterized by a NN.

This yields a fully differentiable model, whose parameters $\boldsymbol{\theta}$ can be learned by gradient descent to maximize Equation 2.2, with a particular algorithm called Backpropagation Through Time (BPTT) (Williams et al. 1995). This algorithm consists of unfolding the RNN by successively processing the inputs $\mathbf{x}_t$ and applying Equation 2.3 with the same parameters $\boldsymbol{\theta}$, given an initial state $\boldsymbol{h}_0$. Gradients are then backpropagated through the successive updates of $\boldsymbol{h}_t$, and accumulated in order to update $\boldsymbol{\theta}$.

The most simple implementation of an RNN, often called *vanilla RNN*, consists of linear transformations and hyperbolic tangent (*tanh*) activations:

$$\boldsymbol{h}_t = tanh(\boldsymbol{W}_x\mathbf{x}_t + \boldsymbol{W}_h\boldsymbol{h}_{t-1} + \boldsymbol{b}_h). \tag{2.3}$$

Here, $\boldsymbol{W}_x \in \mathbb{R}^{d\times n}$ and $\boldsymbol{W}_h \in \mathbb{R}^{d\times d}$ are weight matrices, and $\boldsymbol{b}_h \in \mathbb{R}^d$ is a bias. So in this case, $\boldsymbol{\theta} = \{\boldsymbol{W}_x, \boldsymbol{W}_h, \boldsymbol{b}_h\}$.

While being simple and straightforward, this implementation has a notorious flaw when learned with BPTT: exploding and vanishing gradients (Y. Bengio et al. 1994; Pascanu et al. 2013). Let us look at the derivation of the objective function $\mathcal{L}$ with respect to $\boldsymbol{W}_x$:

$$\frac{\partial \mathcal{L}_T}{\partial \boldsymbol{W}_h} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_T}{\partial \hat{\mathbf{y}}_T} \frac{\partial \hat{\mathbf{y}}_T}{\partial \boldsymbol{h}_T} \frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_t} \frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{W}_h},$$

where $\hat{\mathbf{y}}_T$ is the prediction of the T^{th} output by the model.

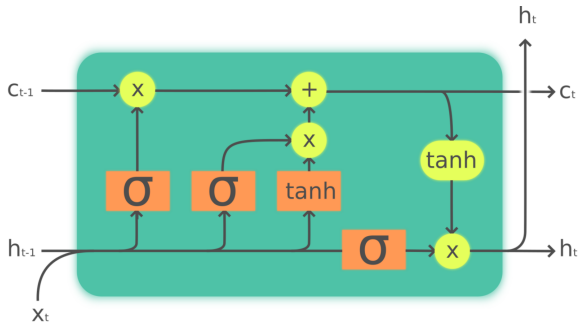


Figure 2.4 – Architecture of an Long Short-Term Memory (LSTM) recurrent network. The gating mechanisms prevent exploding and vanishing gradient. Illustration taken from https://en.wikipedia.org/wiki/Long_short-term_memory.

The issue comes from the $\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_t}$ term that backpropagates the gradient through time and that writes as:

$$\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_t} = \prod_{k=t}^{T} \frac{\partial \boldsymbol{h}_{k+1}}{\partial \boldsymbol{h}_k} = \prod_{k=t}^{T} \left(1 - tanh^2(\boldsymbol{W}_x \mathbf{x}_{k+1} + \boldsymbol{W}_h \boldsymbol{h}_k + \boldsymbol{b}_h)\right) \boldsymbol{W}_h.$$

We can see that the weight matrix $\boldsymbol{W}_h$ is multiplied $T - t$ times by itself. Hence, if at initialization its norm is high, the gradient will grow exponentially with each timestep (exploding gradient), and if it is low, it will decrease exponentially (vanishing gradient). Moreover, the *tanh* activation can also be problematic, as its derivative falls quickly to 0 has its output is moving away from 0. This is depicted in Figure 2.3, where we see that the *tanh* function is easily saturated.

These design issues are tackled by a more sophisticated implementation of RNNs: the Long Short-Term Memory (LSTM) (Hochreiter et al. 1997). The update rule of an LSTM writes as follows:

$$\begin{aligned}
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f[\boldsymbol{h}_{t-1}, \mathbf{x}_t] + \boldsymbol{b}_f) \\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_z[\boldsymbol{h}_{t-1}, \mathbf{x}_t] + \boldsymbol{b}_z) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1}, \mathbf{x}_t] + \boldsymbol{b}_o) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \sigma(\boldsymbol{W}_c[\boldsymbol{h}_{t-1}, \mathbf{x}_t] + \boldsymbol{b}_c) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \sigma(\boldsymbol{c}_t),
\end{aligned}$$

where $[\boldsymbol{h}_{t-1}, \mathbf{x}_t]$ is the concatenation of vectors $\boldsymbol{h}_{t-1}$ and $\mathbf{x}_t$, $\odot$ is the element-wise multiplication between vectors, and σ is the sigmoid function. LSTMs have a second state vector $\boldsymbol{c}_t$, the state of the LSTM, which is controlled by $\boldsymbol{h}_t$ through gating mechanisms. The LSTM is composed of three gates: the forget gate $\boldsymbol{f}_t$, the input gate $\boldsymbol{i}_t$, and the output gate $\boldsymbol{o}_t$. They are called gates because they control the passage of information at certain points in the network. Indeed, since they are activated by a sigmoid function, their values are in $[0, 1]$, and when multiplied to

the states, they let information pass if the value is close to 1, or block it if close to 0. The forget gate can erase information from $\boldsymbol{c}_t$, the input gate controls the information from the input that will affect $\boldsymbol{c}_t$, and the output gate controls the information flow from $\boldsymbol{c}_t$ to $\boldsymbol{h}_t$, which is also the output of the network. This gated mechanism allows them to circumvent the gradient problems of vanilla RNNs by breaking the sequential multiplicative dependency of the gradient.

2.2.2 Training RNNs for Sequence Generation

When training RNNs for sequence generation, inputs and outputs are in the same domain. This raises the question of which input to provide to the RNN while training, observed values, or generated values?

Strictly following the maximum likelihood formulation in Equation 2.2 yields a straightforward answer: the observed values must be fed to the RNN. This method is called Teacher Forcing, as the training algorithm, the "teacher", impose the "right" inputs to the RNN. However, at test time, the RNN is fed with its own outputs, which are not always perfect, and small predictions error can put the RNN in a state unseen during training, leading to larger errors subsequently.

To circumvent this issue, S. Bengio et al. 2015 introduced a curriculum learning approach. During training, when computing the next state $\boldsymbol{h}_t = f_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$, they propose to replace the true value $\boldsymbol{x}_t$ with a value generated by the RNN $\hat{\boldsymbol{x}}_t$ with a probability ϵ, a hyper-parameter of the training algorithm. They also proposed to schedule ϵ from 0 to 1 during training. At the beginning of training, when the RNN has a random behavior, the true values are always fed as inputs, and as training progresses, the RNN is fed more and more with its own outputs. They applied this method to several generation tasks, like image captioning or speech recognition with success. After that, Goyal et al. 2016 proposed the Professor Forcing algorithm, which uses an adversarial algorithm (Goodfellow et al. 2014) to force an RNN to behave in the same when presented with real data or its own inputs.

In their work on conditional video generation, Chiappa et al. 2017 performed a large experimental campaign on the scheduling procedure. They found out that the proportion of teacher forcing employed during training led to a tradeoff between short term and long term prediction. When training is dominated by self-generated inputs, better long term predictions are achieved, but the resulting samples are blurrier. On the other hand, when training is dominated by teacher forcing, the model generates sharper short term predictions, but they deteriorate quickly.

2.2.3 RNN Variants

The LSTM architecture became very popular and is used in most works employing RNNs. But several other implementations were proposed in order to improve or simplify the LSTM architecture.

GRU. The most notable one is the Gated Recurrent Unit (GRU) network (Cho et al. 2014). It fuses some gates in the LSTM to reduce the number of learned parameters, and to use only one state variable. The update rule for the state variable writes as follows:

$$\begin{aligned}
\boldsymbol{z}_t &= \sigma(\boldsymbol{W}_z[\boldsymbol{h}_{t-1}, \mathbf{x}_t] + \boldsymbol{b}_z) \\
\boldsymbol{r}_t &= \sigma(\boldsymbol{W}_r[\boldsymbol{h}_{t-1}, \mathbf{x}_t] + \boldsymbol{b}_r) \\
\boldsymbol{o}_t &= tanh(\boldsymbol{W}_o[\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}, \mathbf{x}_t] + \boldsymbol{b}_r) \\
\boldsymbol{h}_t &= \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) \odot \boldsymbol{o}_t.
\end{aligned}$$

There is still a reset gate $\boldsymbol{r}_t$, but this time it is applied directly to the state variable to compute the new state candidate $\boldsymbol{o}_t$. There is no more input gate, only an update gate $\mathbf{z}_t$, that selects which units will be carried from the previous state, and which will be updated.

IRNN. To further simplify the architecture, Q. V. Le et al. 2015 proposed the Identity RNN. This IRNN is similar to the vanilla RNN, where the *tanh* activation function is replaced by the Rectified Linear Unit (ReLU) activation function, which is 0 when the input is negative, and the identity otherwise. They propose to initialize the update weight matrix $\boldsymbol{W}_h$ as the identity matrix. Combined with ReLU, this means that at the beginning of learning, positive units in $\boldsymbol{h}_t$ are copied to the next state, and negative ones are replaced by 0. Hence, this mechanism emulates the update and forget gate of the LSTM, without learning supplementary parameters. They obtain performances similar or superior to LSTMs on toy tasks, language modeling, and speed recognition. However, because of the popularity of LSTMs, the fact that IRNNs did not achieve any strong performance improvements and where not extensively tested on large scale real-world datasets, it is LSTMs that are still commonly used.

QRNN. Even if RNN architectures are efficient, they still need to process inputs sequentially. It induces a heavy computation burden, especially when sequences become longer, as the process cannot be parallelized. This is why Bradbury et al. 2017 proposed the Quasi-Recurrent Neural Network that relies on Convolutional Neural Networks (CNNs) to improve the parallelization of RNNs.

The different gates and state candidates are computed independently across time by convolution kernels. Hence, the selection of the kernel size affects directly

the temporal range upon which the hidden state is computed. It thus controls the trade-off between computation speed and memory length. The resulting gates and state candidates are then pooled in time with gating function, like classical LSTMs or GRUs. With this framework, authors were able to obtain competitive performances on sentiment classification and language modeling tasks at 3 times the training speed of a similar LSTM on average.

Very active research on RNNs, and particularly LSTMs, architectures led to a large number of possible variants. It then became difficult to choose the right one for a given task. In a large scale experimental campaign, Greff et al. 2017 tested many variations of the LSTM architecture on several tasks and datasets. They first report that no variant is globally better than the others and that the optimal architecture depends on the specific task. They also show that the standard LSTM obtains globally strong performances across all domains, limiting the impact of the diverse variants.

To further study the performances of different RNN architectures, Collins et al. 2017 performed a comparative study of vanilla RNNs, GRUs, and LSTMs. They showed that at a given parameter budget, all three architectures yield similar performances. But they also note that LSTMs tend to converge faster.

All this body of work tends to legitimize the wild use of LSTMs as default architecture to handle sequences in DL tasks.

2.2.4 RNN Based Architectures

RNNs are used as base build blocks in several larger architectures. We present here three popular ones: the multi-layer RNN, the bidirectional RNN, and the sequence to sequence (seq2seq) model.

Multi-Layer RNN

Like many DL architectures, RNNs can be stacked to form multi-layered RNNs. Stacking RNNs yield more powerful models that are able to capture more complex temporal dependencies, but are also more prone to overfitting and need to be regularized carefully.

For instance, stacking two RNNs will yield an activation function of the form:

$$\begin{cases} \boldsymbol{h}_t^1 = f_{\boldsymbol{\theta}^1}(\mathbf{x}_t, \boldsymbol{h}_{t-1}^1), \\ \boldsymbol{h}_t^2 = f_{\boldsymbol{\theta}^2}(\boldsymbol{h}_t^1, \boldsymbol{h}_{t-1}^2), \end{cases}$$

with $\boldsymbol{h}_t^1$ the state of the first layer, $\boldsymbol{h}_t^2$ the state of the second layer, and $\boldsymbol{\theta} = \{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2\}$. In multi-layer RNNs, it is the state of the last layer that is used as input of the final softmax layer.

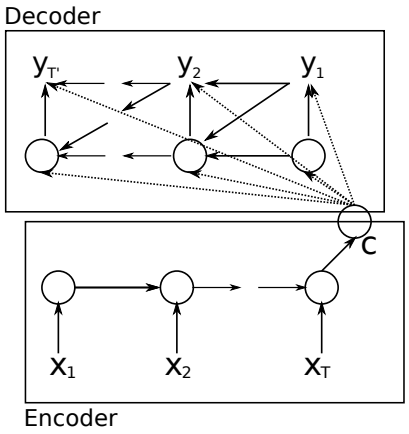


Figure 2.5 – The seq2seq architecture. The encoder encodes the input sequence into a vectorial representation that is used by the decoder to predict the target sequence. The gating mechanisms prevent exploding and vanishing gradient. Illustration taken from Cho et al. 2014.

Bidirectional RNN

In some applications, complete sequences must be encoded. It is particularly the case in NLP, with tasks such as text classification (Howard et al. 2018), named entity recognition (Lample et al. 2016), or part-of-speech tagging (P. Wang et al. 2015).

In such cases, encoding a sequence with a single RNN can be limiting. A solution is to learn two RNNs at the same that parse the sequence in opposite directions. This architecture is called a Bidirectional RNN (BiRNN). One RNN will parse the sequence from left to right, and the other from right to left. To form outputs at each timestep, their states are concatenated. It is also possible to have multi-layer bidirectional RNN. In that case, the input of the next layer is the concatenation of the state of the two RNNs from the last layer.

Sequence to Sequence and Attention

The seq2seq architecture was first proposed by Kalchbrenner et al. 2013 and Cho et al. 2014 for neural machine translation. The model is composed of an encoder RNN and a decoder RNN. In Cho et al. 2014, the encoder is a bidirectional RNN that encodes a source sentence into a vectorial representation $\boldsymbol{c}$. The decoder is an RNN that takes as input $\boldsymbol{c}$ at each timestep in addition to the decoder input and state. Figure 2.5 illustrates the architecture.

In a following work, Bahdanau et al. 2015 proposed to add to the seq2seq architecture an attention mechanism. Instead of conditioning the decoder by the same vector $\boldsymbol{c}$ at each timestep, they proposed to use a linear combination of the encoder states weighted by factors that depend on the decoder current state. The mechanism is illustrated in Figure 2.6.

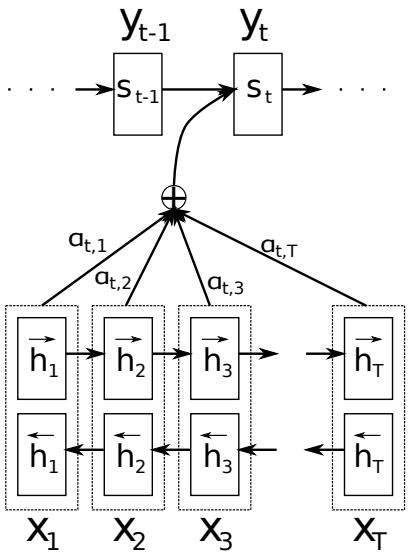


Figure 2.6 – The attention mechanism for the seq2seq model. The decoder is conditioned at time t by a linear combination of the encoder states, weighted by weights $\alpha_{t,i}$ that depend on the decoder state $\boldsymbol{s}_{t-1}$. Illustration taken from Bahdanau et al. 2015.

Formally, let $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T)$ be the vectorial states of the encoder and $\boldsymbol{s}_{t-1}$ the current vectorial state of the decoder. The objective is to compute the conditioning vector $\boldsymbol{c}_t$ that will be used in addition to $\boldsymbol{s}_{t-1}$ to compute the next state $\boldsymbol{s}_t$. This is done as follows:

$$\boldsymbol{c}_t = \sum_{i=1}^{T} \alpha_{t,i} \boldsymbol{h}_i.$$

And the weights $\alpha_{t,i}$ are obtained as follows:

$$\alpha_{t,i} = \frac{\exp(\boldsymbol{e}_{t,i})}{\sum_{k=1}^{T} \exp(\boldsymbol{e}_{t,k})},$$

where $\boldsymbol{e}_{t,i} = a(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i)$ is an attention vector computed by the attention function a. The $\alpha_{t,i}$ weights sum to 1 and the resulting conditioning vector $\boldsymbol{c}_t$ can be viewed as a soft selection of an input state. In the original paper (Bahdanau et al. 2015), this selection mechanism was intended as a way to align the input and output sentences that are from different languages that do not follow the same word positioning rules.

It also helps the learning algorithm by allowing gradient flow between each decoding and encoding states, preventing vanishing and exploding gradients. It was not the case in the original seq2seq model, as gradients had to flow through each output and input states sequentially.

2.2.5 Application: Language Modeling

A direct application of RNNs, and more specifically LSTMs, is language modeling. As an entire chapter of this thesis is dedicated to this task, we present here how the LSTM is applied to language modeling.

Language models are at the heart of numerous works, notably in the text mining and information retrieval communities. Tasks like automatic completion, dialog systems, or automatic translation, are based on language models. Instead of fine-grained semantical analysis, these statistical models aim at extracting word occurrence distributions in different contexts.

Before RNNs, language models were based on unstructured n-grams models. The first breakthrough in neural language modeling was from the C-Bow and skip-gram algorithm proposed in Mikolov et al. 2013. Their algorithm learns word representations by predicting word contexts, i.e. surrounding words in texts. The resulting word representations were then used for many downstream tasks, but often in an unstructured manner.

Language Modeling with RNNs. Meanwhile, RNNs were also used for language modeling. The language modeling problem is easy to formalize in the RNN framework. With $\mathbf{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_T\}$ a text sequence composed of T tokens, the likelihood of the sequence for an RNN with parameters θ writes as follows:

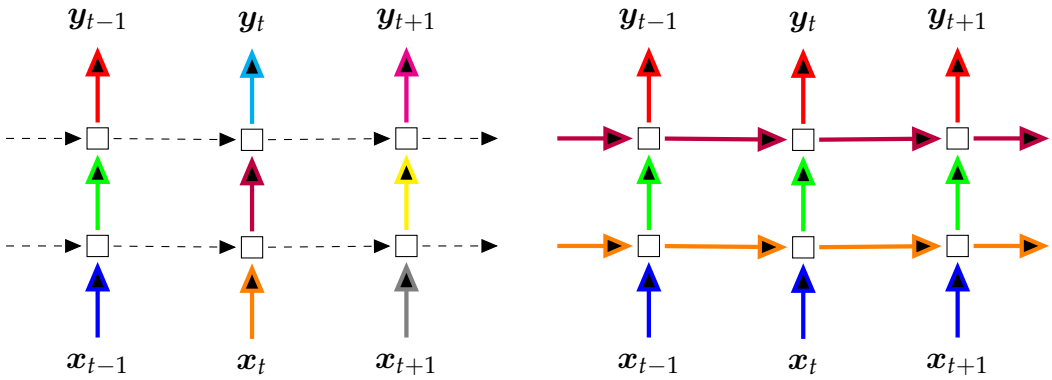
$$p_\theta(\mathbf{x}) = \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_t | \boldsymbol{x}_{1:t-1}) = \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_t | \boldsymbol{h}_t),$$

where $\boldsymbol{x}_0$ is a token symbolizing the start of the sequence, and $\boldsymbol{h}_t$ is the latent state of the RNN. $\boldsymbol{x}_t$ are discrete tokens, hence they cannot be directly fed into the LSTM. Instead, we learn an embedding matrix $\boldsymbol{U}$, which is a lookup table that matches a token with a continuous vector. Since the outputs of the model are discrete token, $p_\theta(\boldsymbol{x}_t | \boldsymbol{h}_t)$ corresponds to a categorical distribution obtained by the application of a softmax layer on top of the output of the RNN:

$$p_\theta(\boldsymbol{x}_t | \boldsymbol{h}_t) = \frac{e^{\boldsymbol{h}_t \top \boldsymbol{v}_{x_t}}}{\sum_{\boldsymbol{x} \in V} e^{\boldsymbol{h}_t \top \boldsymbol{v}_{\boldsymbol{x}}}},$$

where $\boldsymbol{v}_{\boldsymbol{x}}$ is a vector of learnable parameters corresponding to token $\boldsymbol{x}$, and V is the vocabulary.

In the last few years, several works proposed modifications of the LSTM architecture to learn better language models (Merity et al. 2017; Zilly et al. 2016). However, Merity et al. 2018b and Melis et al. 2018 concurrently show that a vanilla LSTM, with careful tuning and a few regularization techniques, can achieve state-of-the-art language modeling performances. We present here two important regularization techniques, weight tying and dropout, and how they are adapted for the task.



(a) Naive dropout RNN (b) Variational Dropout RNN

Figure 2.7 – Variational dropout technique (Gal et al. 2016) (right) compared to the standard technique (left). Each square represents an RNN unit, with horizontal arrows representing time dependence (recurrent connections). Vertical arrows represent the input and output to each RNN unit. Coloured connections represent dropped-out inputs, with different colours corresponding to different dropout masks. On the left, all masks are different at each timestep. On the right, the same masks are used at each timestep. Dashed lines correspond to standard connections with no dropout. Illustration taken from Gal et al. 2016.

Weight Tying. A first regularization technique is weight tying (Inan et al. 2017): sharing the weights between the word embeddings and the decoder matrices. This means that for all word embedding $\boldsymbol{u_x} \in \boldsymbol{U}$, $\boldsymbol{u_x} = \boldsymbol{v_x}$. Those matrices are usually very large because they scale linearly with the number of individual tokens. Hence, sharing these matrices reduce drastically the number of parameters of the model.

Dropout. Dropout is one of the main neural network regularization techniques (Hinton et al. 2012). The principle is to randomly mask a portion of layers' activations, preventing co-adaptations of neurons. This encourages the network to produce redundant outputs, but never identical. With this technique, larger layers can be used without overfitting, globally enhancing results.

In RNNs, dropout can be applied at several spots in the network: on the embedding matrix, between RNN layers, before the softmax layer. Gal et al. 2016 proposed an implementation of dropout specific to RNN: variational dropout, depicted in Figure 2.7. In this method, dropout masks are kept constant on the entire sequence, offering a more theoretically grounded framework (Gal et al. 2015), and better results on language modeling. In the same paper, they also propose to apply dropout at the word level, by masking a portion of the conditioning

tokens of the LSTM. This prevents the LSTM from learning hard assignments in the presence of rare-words.

2.3 Other Deep Temporal Models

One of the principal challenges of sequential tasks is to handle long-term dependencies. We present here models specifically designed to augment RNNs memories.

2.3.1 Convolutional Neural Network

Convolutional Neural Networks are NNs designed to handle discrete signals and in particular images. Vectorial time series can be seen as 1D signals, and therefore 1D convolutions can be applied.

Let $\mathbf{x}$, with $\mathbf{x}_t \in \mathbb{R}^n$ an input sequence of size T. A 1D-CNN is composed of a kernel $\boldsymbol{\theta} \in \mathbb{R}^{d \times k \times n}$, where d is the output dimension and k the kernel size. This kernel is a tensor of learnable parameters that are used to compute the next state $\boldsymbol{h}_t$ as follows:

$$\boldsymbol{h}_t^i = \sum_{j=1}^{k} < \mathbf{x}_{t-j}, \boldsymbol{\theta}_{i,j} >, \tag{2.4}$$

where $\boldsymbol{h}_t^i$ is the i^{th} component of $\boldsymbol{h}_t$, and $\boldsymbol{\theta}_{i,j} \in \mathbb{R}^n$ is the vector a position (i, j) in $\boldsymbol{\theta}$. By applying Equation 2.4 sequentially on the whole sequence, an output representation is formed. Like other DL architectures, CNNs can be stacked on top of each other to form deep neural networks. This formulation was first proposed in X. Zhang et al. 2015 and applied to character-based text classification.

The principal advantage of this architecture is that all states $\boldsymbol{h}_t$ at all times t can be computed in parallel as they depend on their input only. However, the main drawback is that the model can only look at the k last inputs to compute the state, which prevents long-term reasoning.

2.3.2 Memory Network

One of the first attempts to extend the memory of RNNs was the Memory Network (Sukhbaatar et al. 2015). While RNNs can theoretically retain information at arbitrary long-range, in practice, they tend to have a relatively short memory, and "forget" information after a few dozen of timesteps. That is why some techniques like continuous cache pointers (Grave et al. 2017a), that biased the output of trained language models toward words that appeared previously in the text, work well. For instance, in the Wikipedia corpus, the word "jaguar" does not

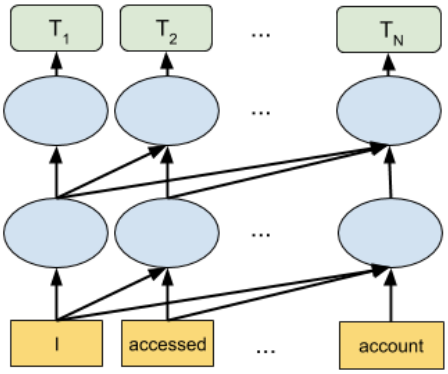


Figure 2.8 – The transformer network for language modeling. The blue ovals are hidden stats. We can see that each hidden state is computed by attending at all past states generated by the previous layer. Illustration taken from https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

appear often in a random article. But if it appeared once, the current article is probably about jaguars, and the word is more likely to appear again.

Researchers tried to equip RNNs with a separate memory. In the Memory Network, the memory takes the form of a matrix where each row corresponds to a piece of stored information. An RNN can "read" and "write" in this memory with an attention mechanism (Bahdanau et al. 2015). This model obtained state-of-the-art results on several language comprehension tasks

2.3.3 Transformer Network

Taking the attention mechanism to its limits, Vaswani et al. 2017 proposed an attention-only model. While RNNs states are computed recursively by incorporating inputs one after the other, the transformer computes its state with an attention mechanism on all previous inputs, see Figure 2.8. With this architecture, they attain state-of-the-art results on the machine translation task.

Following this breakthrough, many researchers proposed extensions on the model, primarily in the Natural Language Processing (NLP) community. Devlin et al. 2019 proposed the BERT model, where they use a transformer to pre-train an Language Model (LM) on variations of the self-supervised language modeling task. By then fine-tuning the model on downstream tasks, they beat state-of-the-art results on many NLP tasks like question answering or sentiment analysis. Several adaptations and improvements were proposed, for instance in Radford et al. 2019 or Dai et al. 2019, or for other languages, like French (H. Le et al. 2019) for instance.

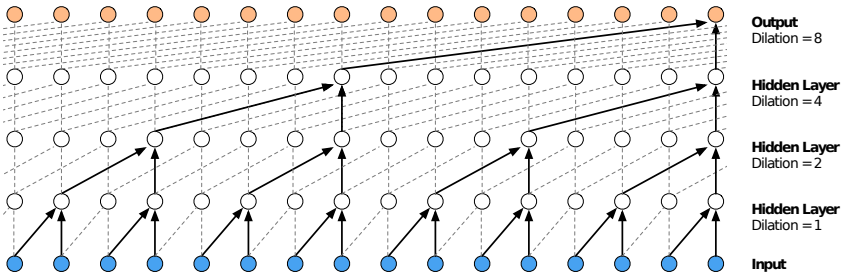


Figure 2.9 – The WaveNet architecture. The dilated convolutional architecture gives a large receptive field to the network while limiting the total number of parameters and computations. Illustration taken from Oord et al. 2016a

The success of the model in NLP inspired researchers in other domains where data are sequential. A recent example is Weissenborn et al. 2019 that proposed a spatio-temporal version of the transformer network for video prediction.

2.3.4 WaveNet

Another research direction on temporal models is based on CNNs. A particularly successful contribution in this direction is the WaveNet (Oord et al. 2016a). This model was designed to handle audio signals, that have the particularity of containing simultaneously very high and very low frequencies. Audio sequences are usually long, as a very high sampling frequency is required to produce high-definition audio. It also means that models must cover a large period, in terms of the number of timesteps, to account for very low frequencies.

To account for all these challenges, they proposed to use dilated convolutions, as pictured in Figure 2.9. This kind of convolutional kernel has a stride equal to the kernel size, reducing the number output feature vectors.

With notations of Equation 2.4, the update rule would be:

$$\boldsymbol{h}_{t+1}^{i} = \sum_{j=1}^{k} < \mathbf{x}_{t-k*(j-1)}, \boldsymbol{\theta}_{i,j} > .$$

By stacking such layers, high-level output vectors cover a large, but less dense, temporal span, while lower cover a smaller, but denser, temporal span. Hence, the architecture can work on a large range of frequencies, while keeping the number of parameters reasonable. They were thus able to generate high-definition speech and music.

2.3.5 Physics Based Models

Physical processes represent a large part of temporal data. Physicists design mathematical systems to infer unobserved states of such processes and predict future values. These models are the fruit of long research history and have proven their efficiency. That is why several DL researchers proposed deep dynamic models guided by the physical and mathematical models, such as Bézenac et al. 2018 that introduced a neural network derived from partial differential equations to predict sea surface temperature.

Learning the parameters of differential equations based models with neural networks is a recent research direction in DL (Y. Lu et al. 2018; Long et al. 2018). It is based on the observation that Residual Neural Networks (ResNets) (K. He et al. 2016) can be seen as Euler discretization of continuous transformations. A ResNet is a network composed of a succession of residual blocks. A residual block in a neural network which outputs are added to its inputs, as follows:

$$\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + f(\boldsymbol{h}_t, \theta),$$

where $\boldsymbol{h}_t$ is the input at step t of the block and f the residual neural network with parameters θ.

In standard deep dynamical models, the step size is fixed, and models are discrete in time. But Chen et al. 2018 observed that if the step size is reduced, the limit is a continuous dynamics that can be parameterized by an Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\boldsymbol{h}(t)}{\mathrm{d}t} = f(\boldsymbol{h}(t), t, \theta).$$

They thus proposed a framework for learning such networks with black-box differential equation solver, producing ODE based neural models that are continuous in time. They used it to learn continuous dynamics of physical processes like Lorenz Attractor.

2.4 Sequential VAEs

Another family of dynamic sequential models is based on deep variational methods. Rather than learning point representations $\boldsymbol{h}_t$, these generative models learn to infer distributions of latent variables $\mathbf{z}_t$. The latent space is shaped by a simple prior, typically a standard Gaussian, that is easy to sample from. Modeling latent representations in a probabilistic manner allows the model to capture uncertainties in the data. And temporal data often present uncertainties. For instance, uncertainties in the future, or uncertainties on unobserved quantities in complex systems like climate.

2.4.1 The Variational Auto-Encoder

Let $\mathbf{z} \in \mathbb{R}^d$ be the latent variable that generates observation $\mathbf{x} \in \mathbb{R}^n$. Inferring latent continuous variables in a Bayesian framework is hard in the general case, as the posterior distribution $p(\mathbf{z}|\mathbf{x})$ is often intractable. One way to circumvent this issue is by approximating it by a so-called variational distribution $q(\mathbf{z})$. The goal is then to find the variational distribution that is the closest to $p(\mathbf{z}|\mathbf{x})$ by minimizing their Kullback-Leibler Divergence (KLD), that writes as follows:

$$D_{\mathrm{KL}}\left(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})\right) = \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \, \mathrm{d}\mathbf{z}.$$

By Bayes's rule, we can write:

$$\begin{aligned} D_{\mathrm{KL}}\left(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})\right) &= \int q(\mathbf{z}) \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})} + \log p(\mathbf{x})\right] \mathrm{d}\mathbf{z} \\ &= \int q(\mathbf{z}) \left[\log q(\mathbf{z}) - \log p(\mathbf{z},\mathbf{x})\right] \mathrm{d}\mathbf{z} + \log p(\mathbf{x}). \end{aligned}$$

In the last line, the marginal likelihood of the data $p(\mathbf{x})$ appears. This quantity depends only on the data. We can rearrange the equation as follows:

$$\begin{aligned} \log p(\mathbf{x}) &= D_{\mathrm{KL}}\left(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})\right) + \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{z},\mathbf{x}) - \log q(\mathbf{z})\right] \\ &= D_{\mathrm{KL}}\left(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})\right) + \mathcal{L}(q). \end{aligned}$$

As $\log p(\mathbf{x})$ does not vary with respect to q, maximizing $\mathcal{L}(q)$ leads to the minimization of $D_{\mathrm{KL}}\left(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})\right)$, which is our primary goal. And since the KLD is non-negative, $\mathcal{L}(q)$ is actually a lower bound of the marginal likelihood, or evidence, of the data, called the Evidence Lower Bound (ELBO).

So, the goal of variational inference is to maximize

$$\begin{aligned} \mathcal{L}(q) &= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{z},\mathbf{x}) - \log q(\mathbf{z})\right] = \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z})\right] \\ &= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \mathbb{E}_{q(\mathbf{z})}\left[\log q(\mathbf{z}) - \log p(\mathbf{z})\right] \\ &= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - D_{\mathrm{KL}}\left(q(\mathbf{z})||p(\mathbf{z})\right). \end{aligned}$$

In simple cases, the variational distribution can be computed with a coordinate ascent algorithm. However, we are interested here in large datasets of high-dimensional and structured data, such as images for example. Hence, to produce qualitative samples, we use a NN for the likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, where $\boldsymbol{\theta}$ are the parameters of the NN. Moreover, with large datasets, learning the parameters of all the variational distribution $q(\mathbf{z})$ for all data points becomes intractable. A strategy is to perform amortized Variational Inference (VI) by learning a NN, named a recognition network, to predict the parameter of the posterior given a

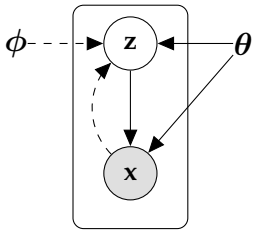


Figure 2.10 – Variational Auto-Encoder (Kingma et al. 2014). The grey circle represents latent variables and the white circle observed variables. The panel represents the dataset ensemble Full lines represent the generative model and dashed lines the inference model.

particular observation $\mathbf{x}$: $q_\phi(\mathbf{z}|\mathbf{x})$, where ϕ are the parameters of the recognition model. The corresponding graphical modeling is depicted in Figure 2.10, and the ELBO that is maximized now writes as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right). \tag{2.5}$$

The challenge here is the to jointly learn the neural network parameters $\boldsymbol{\theta}$ and ϕ, who would give an expressive model, but cannot be learned by traditional EM methods. The goal is then to learn the full model by gradient descent, but there is an issue with the gradient of ϕ with respect to the ELBO. Indeed, the likelihood term is an expectation over $q_\phi(\mathbf{z}|\mathbf{x})$ that has no closed-form since we use neural networks. We hence have to estimate it empirically, which requires to sample from $q_\phi(\mathbf{z}|\mathbf{x})$, breaking the gradient flow from the ELBO.

To do this, Kingma et al. 2014 and Rezende et al. 2014 concurrently proposed a simple but efficient method: the reparametrization trick. By choosing $q(\mathbf{z})$ adequately, it is possible to reparametrize the sample function, allowing the computation of an unbiased and low variance estimation of the gradient. In practice, and in the rest of this manuscript, $q(\mathbf{z})$ is a Gaussian distribution with a diagonal covariance matrix. In this case, the reparameterized sampling procedure $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma^2(\mathbf{x})I)$, where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are the NNs forming the recognition network $q_\phi(\mathbf{z}|\mathbf{x})$, is:

$$\mathbf{z} = g_\phi(\mathbf{x}, \epsilon) = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, I)$. With this reparametrization, the only stochastic component is ϵ, which does not depend on ϕ, and can be viewed as an input of the model.

We just saw how to compute the likelihood term of the ELBO, and how to compute the gradient on both ϕ and $\boldsymbol{\theta}$, and the only term left is the KLD between the posterior and the prior. By making the prior Gaussian $p(\mathbf{z}) = \mathcal{N}(\mu_p, \sigma_p^2 I)$, it is possible to derive a closed-form from this KLD:

$$D_{\mathrm{KL}}\left(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right) = \log \frac{\sigma_p}{\sigma(\mathbf{x})} + \frac{\sigma^2(\mathbf{x}) + (\mu_p^2 - \mu^2(\mathbf{x}))}{2\sigma_p^2} - \frac{1}{2}.$$

Figure 2.11 – Manifold learned on the Frey Faces dataset by a VAE in Kingma et al. 2014, where the latent space is in 2D. The four images in the corners are real images, whose latent representations were inferred by the recognition model. The resulting latent codes are then interpolated on a grid, and the decoded images are displayed on the figure. We can see that transitions in the latent space are smooth, indicating that the latent space organizes the data semantically. Illustration taken from Kingma et al. 2014.

We just described the full Variational Auto-Encoder (VAE). The model can be viewed as a regularized auto-encoder, where $q_\phi(\mathbf{z}|\mathbf{x})$ is an encoder, and $p_\theta(\mathbf{x}|\mathbf{z})$ a decoder. In this sense, the likelihood term in the ELBO can be seen as a reconstruction term and the KLD as a regularization term. Usually, auto-encoders are regularized by adding an ℓ_2 norm on the network parameters, whereas here it is the latent codes that are regularized. The KLD term pushes all variational distribution to be close from the prior, while the likelihood term encourages good reconstructions.

The VAE is one of the first contribution to the Bayesian Deep Learning subfield. It offers a principally Bayesian framework to design NNs. The VAE is also a powerful generative model, as it is possible to generate samples from the posterior and the prior. Moreover, it empirically appears that VAEs learn a semantic latent space, as can be seen on the latent manifold visualization in Figure 2.11. Further research in this direction tends to confirm this capacity of the VAE to learn disentangled representations of data (Higgins et al. 2017), where latent dimensions are independent of one another, which find applications speech recognition (Hsu et al. 2017) for instance.

2.4.2 Autoregressive Models

The VAE framework was developed and extended in many ways, and in particular for sequential data, with applications such as sequence modeling, generation,

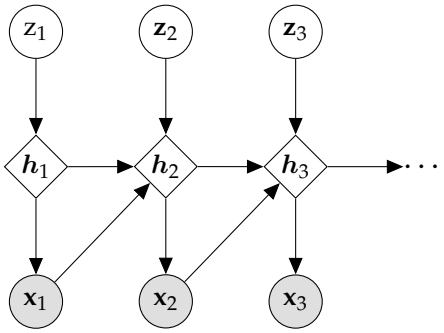


Figure 2.12 – Stochastic Recurrent Network (Bayer et al. 2014).

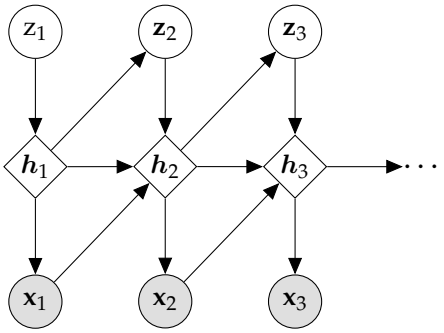


Figure 2.13 – Variational Recurrent Neural Network (Chung et al. 2015).

and prediction. When using RNNs for sequence generation, for instance, the sampling process appends in the observation space, which can be a problem if it is multi-modal and in high dimension. By instead using deep Variational Inference (VI), the sampling process can be performed in a more controlled latent space, at the cost of approximate likelihood maximization.

One of the first VAE extensions to sequential tasks was proposed by Bayer et al. 2014. They straightforwardly augment an RNN with random variables inferred by deep VI, where $q_\phi(\mathbf{z}|\mathbf{x})$ is parameterized by a forward RNN. The graphical model is pictured in Figure 2.12. In this model, the random states $\mathbf{z}_t$ are independent in time and are used to add stochasticity into a classical RNN. The resulting lower bound is the same as in Equation 2.5. The method was used on multivariate time series: polyphonic music and motion capture.

Afterward, Chung et al. 2015 modified the model by adding structure between the latent stochastic states, as can be seen in Figure 2.13. This added temporal structure allows the model to put more semantic information into the latent space. Indeed, in Bayer et al. 2014, since the latent states are independent in time, the same state can be sampled from $p_{\boldsymbol{\theta}}(\mathbf{z}_t)$ for radically different $\boldsymbol{h}_t$. The RNN hence has the additional task of interpreting potentially identical latent state in a different way depending on the current state $\boldsymbol{h}_t$.

In contrast, the generative model of Chung et al. 2015 is structured in time through the RNN latent state: $\mathbf{z}_t \sim p_{\boldsymbol{\theta}}(\mathbf{z}_t|\boldsymbol{h}_{t-1})$. Hence, the latent state $\mathbf{z}_t$ carries temporal information, more easily interpretable by the RNN. The stronger decoupling between the temporal stochasticity, encapsulated in the latent space, and the temporal dynamic, learned by the RNN, allows the model to obtain better performances on the same tasks.

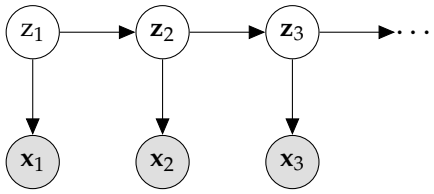


Figure 2.14 – Deep Markov Model (Krishnan et al. 2017).

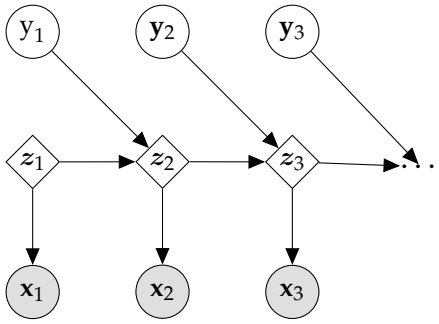


Figure 2.15 – Deep Variational Bayes Filter (Karl et al. 2017).

2.4.3 State Space Models

RNNs are autoregressive models, which have the drawback of accumulating errors when performing long term prediction, as described in Section 2.1.3. Moreover, RNNs are usually trained by taking real data as input. However, when generating future data, they are fed with their own prediction, which can lead to computational instabilities. Autoregressive models also come with high computational costs when used for high dimensional data, such as videos, since each frame as to be re-encoded.

That is why several works explored SSMs learned with deep VI. Krishnan et al. 2015; Krishnan et al. 2017 proposed a deep non-linear SSM using Deep Neural Networks (DNNs) as recognition and dynamic model. The corresponding graphical model is pictured in Figure 2.14. They propose to train the model with the following ELBO:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{t=1}^{T} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{z}_t)\right] - D_{\mathrm{KL}} \left(q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}_1)\right) - \sum_{t=2}^{T} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{x})} \left[D_{\mathrm{KL}} \left(q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}_t|\mathbf{z}_{t-1})\right)\right]. \quad (2.6)$$

The main difference between Equation 2.6 and the ELBO in Equation 2.5 is in the third term. The KLD is now an expectation over the inferred previous latent state $\mathbf{z}_{t-1}$. This allows parallel computation of the KLD terms in the sum but breaks the temporal flow of gradient.

It follows that the design of the inference network $q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{x})$ becomes crucial to stabilize the model. In Krishnan et al. 2017 they explore several architectures and show that designing $q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{x})$ as a bidirectional RNNs on the entire input sequence yield better performances.

The main drawback of this formulation is that the dynamics $p_{\boldsymbol{\theta}}(\mathbf{z}_t|\mathbf{z}_{t-1})$ is only learned with these KLD terms, and we saw there is no gradient flow through time.

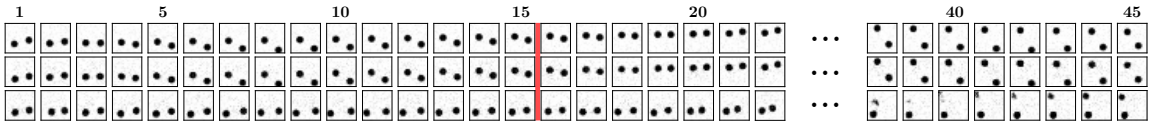


Figure 2.16 – Ground truth (top), reconstructions (middle), generative samples (bottom) from identical initial latent states for the two bouncing balls experiment. The red bar indicates the length of the training sequences, showing that the model is able to extrapolate beyond the length seen during training. Illustration taken from Karl et al. 2017.

Hence, the dynamics is learned independently on timestep pairs. Watter et al. 2015 mitigate this issue by retropopagating the likelihood through the dynamics prior function. This adds a supplementary loss on the dynamics, but gradients are still computed on timestep pairs. Also, the resulting loss is no longer a proper lower bound.

Going further, Karl et al. 2017 proposed to learn a deterministic dynamics that depend on random independent latent states, pictured in Figure 2.15. This yields a deep SSM, learned with a proper ELBO, where gradients can flow through the whole sequence thanks to the deterministic states $\mathbf{z}_t$ while handling stochasticity in the auxiliary random state $\mathbf{y}_t$. They hence were able to show state-of-the-art long term stochastic prediction on toy video datasets, like bouncing balls and pendulums, a sample of which can be seen in Figure 2.16.

CHAPTER 3

SPATIO-TEMPORAL NEURAL NETWORKS

Contents

Chapter abstract

In this chapter, we are interested in spatio-temporal data, which are multivariate time series that are correlated spatially. Contrary to current deep learning methods, like Recurrent Neural Networks (RNNs), that handle these correlations implicitly, we propose a framework to explicitly model these relations. Our method shows superior results on prediction and imputation tasks on several datasets. Experiments show the ability of the approach to extract relevant spatial relations.

- Ali Ziat*, Edouard Delasalles*, Ludovic Denoyer, and Patrick Gallinari (2017). "Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery". In: *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pp. 705–714.

- Edouard Delasalles, Ali Ziat, Ludovic Denoyer, and Patrick Gallinari (2019c). "Spatio-temporal neural networks for space-time data modeling and relation discovery". In: *Knowledge and Information Systems* 61.3, pp. 1241–1267.

3.1 Introduction

Time series exhibiting spatial dependencies are present in many domains including ecology, meteorology, biology, medicine, economics, traffic, and vision. The observations can come from multiple sources e.g. GPS, satellite imagery, video cameras, etc. Several difficulties arise when modeling spatio-temporal data, among them: 1) their size: sensors can cover very large space and temporal lags; 2) the complexity of the underlying generation process, which might be highly non-linear; and 3) the inherent uncertainty of the measurements: sensors are not perfect, and data points are frequently missing or noisy. Answering these challenges, i.e. reducing the spatial dimensionality, uncovering the underlying data generation process, and modeling data uncertainty naturally leads to considering latent dynamic models. This has been exploited both in statistics (Cressie et al. 2011) and in Machine Learning (ML) (Bahadori et al. 2014; Koppula et al. 2013).

Deep Learning (DL) has also developed a large range of dynamic models that are able to capture meaningful features of the sequential data generation processes. However, DL models for structured data are usually restricted to videos, such as the Convolutional RNN (X. Shi et al. 2015; Srivastava et al. 2015) or video pixel networks (Kalchbrenner et al. 2017), for instance. One of the first DL models for graph data was proposed in Kipf et al. 2017, which is concurrent with this work.

We introduce a general class of deep spatio-temporal models for time series of spatial processes. They allow us to explicitly model both spatial and temporal dependencies. The model is designed to capture the dynamics and correlations in multiple series at the spatial and temporal levels. This is a dynamical system model with two components: one for capturing the spatio-temporal dynamics of the process into latent states, and one for decoding these latent states into actual series observations. The model is tested and compared to state-of-the-art alternatives, including Recurrent Neural Networks (RNNs), on several datasets for imputation and forecasting tasks. Tests were performed on time series coming from various domains: health, traffic, meteorology, and oceanography. Besides a quantitative evaluation on forecasting and imputation tasks, the ability of the model to discover relevant spatial relations between series is also analyzed.

The chapter is organized as follows: the model is presented for the forecasting task in sections 3.2 and 3.3 with its different variants, and for the imputation task in section Section 3.4. The experiments are described in Section 3.5 for forecasting (3.5.2), relations discovery (3.5.3) and imputation (3.5.4).

3.2 The Spatio-Temporal Neural Network Model

3.2.1 Notations and Task

Spatio-temporal time series we consider are multivariate and can be high-dimensional. We thus adopt formalism slightly different from the one described in Section 2.1. We consider sets of n temporal series, with m dimensions and of length T. Hence, $m = 1$ means that we consider n univariate series, while $m > 1$ corresponds to n multivariate series each with m components. $\boldsymbol{X}$ is the value history of length T available for training. $\boldsymbol{X}$ is then a tensor in $\mathbb{R}^{T \times n \times m}$, such that $X_t^i \in \mathbb{R}^m$ is an m-dimensional vector containing values of series i at time t. $\boldsymbol{X}_t$ will denote the slice of $\boldsymbol{X}$ at time t, such that $\boldsymbol{X}_t \in \mathbb{R}^{n \times m}$ denotes the values of all the series at time t.

We consider two tasks: forecasting and imputation. For simplicity, we first present the forecasting model in a mono-relational setting. An extension to multi-relational series where different relations between series are observed is described in Section 3.2.4. We consider that the spatial organization of the sources is captured through a matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$. Ideally, $\boldsymbol{W}$ would indicate the mutual influence between sources. In practice, it might be a proximity or similarity matrix between the sources: for geospatial problems, this might correspond to the inverse of a physical distance - e.g. geodesic - between sources. For other applications, this might be provided through local connections between sources using a graph structure (e.g. adjacency matrix for connected roads in a traffic prediction application or graph kernel on the web). Firstly, we make the hypothesis that $\boldsymbol{W}$ is provided as a prior on the spatial relations between the series. An extension where these relations are learned is presented in Section 3.3.

We then consider in the remainder of this section the problem of spatial time series forecasting. We want to learn a model $f : \mathbb{R}^{T \times n \times m} \times \mathbb{R}^{n \times n} \to \mathbb{R}^{\tau \times n \times m}$ able to predict τ timesteps into the future based on $\boldsymbol{X}$ and on their spatial dependency.

3.2.2 Modeling Time Series with Continuous Latent Factors

Let us first introduce the model in the simpler case of multiple time series prediction, without considering spatial relations. The model has two components.

The first one captures the dynamic of the process and is expressed in a latent space. Let $\boldsymbol{Z}_t \in \mathbb{R}^{n \times d_z}$ be the latent representation, or latent factors, of the series at time t. The dynamical component writes $\boldsymbol{Z}_{t+1} = g(\boldsymbol{Z}_t)$. The second component is a decoder which maps latent factors $\boldsymbol{Z}_t$ onto a prediction of the actual series values at t: $\tilde{\boldsymbol{X}}_t = d(\boldsymbol{Z}_t)$, $\tilde{\boldsymbol{X}}_t$ being the prediction computed at time t.

Learning Problem. The objective is to learn the two mapping functions d and g together with the latent factors $\boldsymbol{Z}_t$, directly from the observed series. We formalize this learning problem with a bi-objective loss function that captures the dynamics of the series in the latent space and the mapping from this latent space to the observations. Let $\mathcal{L}(d, g, Z)$ be this objective function:

$$\mathcal{L}(d, g, Z) = \frac{1}{T}\sum_t \Delta(d(\boldsymbol{Z}_t), \boldsymbol{X}_t) + \lambda \frac{1}{T}\sum_{t=1}^{T-1} ||\boldsymbol{Z}_{t+1} - g(\boldsymbol{Z}_t)||^2. \tag{3.1}$$

The first term of the right-hand side of Equation 3.1 measures the ability of the model to reconstruct the observed values $\boldsymbol{X}_t$ from the latent factor $\boldsymbol{Z}_t$. It is based on loss function Δ which measures the discrepancy between predictions $d(\boldsymbol{Z}_t)$ and ground truth $\boldsymbol{X}_t$. The second term aims at capturing the dynamics of the series in the latent space. This term forces the system to learn latent factors $\boldsymbol{Z}_{t+1}$ that are as close as possible to $g(\boldsymbol{Z}_t)$. The hyper-parameter λ is used here to balance this constraint and is fixed by cross-validation. The solution d^*, g^*, Z^* to this problem is computed by minimizing $\mathcal{L}(d, g, Z)$:

$$d^*, g^*, \boldsymbol{Z}^* = \arg\min_{d,g,\boldsymbol{Z}} \mathcal{L}(d, g, \boldsymbol{Z}). \tag{3.2}$$

Learning Algorithm. In the presented setting, functions d and g, described in the next section, are differentiable parametric functions. Hence, the learning problem can be solved end-to-end with Stochastic Gradient Descent (SGD) techniques [1] directly from Equation 3.2. At each iteration, a time t is sampled, and $\boldsymbol{Z}_t$, $\boldsymbol{Z}_{t+1}$, g, and d are updated according to the gradient of Equation 3.1. Training can also be performed via mini-batch, resulting in a high learning speed-up when using Graphical Processing Units (GPUs) which are the classical configuration for running such methods. The details of the mini-batched learning algorithm can be found in Algorithm 3.1.

Prediction. Once the model is learned, it can be used to predict future values of the series as follows: the latent factors of any future state of the series are computed using the g function, and the corresponding observations are predicted by using d on these factors. Formally, let us denote $\tilde{\boldsymbol{Z}}_\tau$ the predicted latent factors at time $T + \tau$. The forecasting process computes $\tilde{\boldsymbol{Z}}_\tau$ by successively applying the g function τ times on the learned vector $\boldsymbol{Z}_T$:

$$\tilde{\boldsymbol{Z}}_\tau = g \circ g \circ ... \circ g(\boldsymbol{Z}_T) = g^{(\tau)}(\boldsymbol{Z}_T),$$

and then computes the predicted outputs : $\tilde{\boldsymbol{X}}_\tau = d(\tilde{\boldsymbol{Z}}_\tau)$

1. In the experiments, we used the Nesterov's Accelerated Gradient (NAG) method (Sutskever et al. 2013).

3.2.3 Modeling Spatio-Temporal Series

Algorithm 3.1 Learning algorithm with mini-batches

Inputs: Dataset $\boldsymbol{X}_t \ \forall t \in \{1, ..., T\}$, number of iterations E, min-batch size B.
Parameters: decoder d, dynamics g, latent states $\boldsymbol{Z}_t \ \forall t \in \{1, ..., T\}$
for $e = 1 \rightarrow E$ **do**
  $grad \leftarrow 0$
  for $k = 1 \rightarrow B$ **do** ▷ Can be parallelized as iterations are independent.
    Sample a series i and a timestep t uniformly
    Compute $\hat{Z}_t^i \leftarrow h(Z_{t-1}^i \Theta^{(0)} + \boldsymbol{W}^i \boldsymbol{Z}_{t-1} \Theta^{(1)})$
    Compute $\hat{X}_t^i \leftarrow d(Z_t^i)$
    Compute loss $\mathcal{L}_{e,k} = \Delta(\hat{X}_t^i, X_t^i) + \lambda ||Z_t^i - \hat{Z}_t^i||^2$
    Accumulate gradient: $grad \leftarrow grad + \nabla \mathcal{L}_{e,k}$
  end for
  Update parameters using gradient descent with $grad$
end for

Let us now introduce a spatial component in the model. We consider that each series has its own latent representation at each timestep. $\boldsymbol{Z}_t$ is thus a $n \times d_z$ matrix such that $Z_{t,i} \in \mathbb{R}^{d_z}$ is the latent factor of series i at time t, d_z being the dimension of the latent space. It is this spatial component that distinguishes the proposed model from the classical RNN or the Dynamic Factor Graph from Mirowski et al. 2009 presented in Section 2.1.3. Indeed, in both those models, $\boldsymbol{Z}_t$ is a single vector common to all the series.

The spatial information is integrated into the dynamic component of the model through a matrix $\boldsymbol{W} \in \mathbb{R}_+^{n \times n}$, with n the number of sources. The latent representation of any series at time $t+1$ depends on its own latent representation at time t (intra-dependency) and on the latent representations of the other series at t (inter-dependency) through the dynamic model $g(\boldsymbol{Z}_t)$:

$$\boldsymbol{Z}_{t+1} = g(\boldsymbol{Z}_t) = h(\boldsymbol{Z}_t \Theta^{(0)} + \boldsymbol{W} \boldsymbol{Z}_t \Theta^{(1)}),$$

where $\Theta^{(0)} \in \mathbb{R}^{d_z \times d_z}$ and $\Theta^{(1)} \in \mathbb{R}^{d_z \times d_z}$ are linear mappings, and h is a non-linear function. In the experiments we set $h = tanh$ but h could also be a more complex parameterized function like a Multi-Layer Perceptron (MLP) for instance. The resulting optimization problem over d, $\boldsymbol{Z}$, $\Theta^{(0)}$, and $\Theta^{(1)}$ writes:

$$\begin{aligned} d^*, \boldsymbol{Z}^*, \Theta^{(0)*}, \Theta^{(1)*} = \underset{d, \boldsymbol{Z}, \Theta^{(0)}, \Theta^{(1)}}{\arg\min} & \frac{1}{T} \sum_t \Delta(d(\boldsymbol{Z}_t), \boldsymbol{X}_t) \\ & + \lambda \frac{1}{T} \sum_{t=1}^{T-1} ||\boldsymbol{Z}_{t+1} - h(\boldsymbol{Z}_t \Theta^{(0)} + \boldsymbol{W} \boldsymbol{Z}_t \Theta^{(1)})||^2. \end{aligned} \tag{3.3}$$

Algorithm 3.1 details the mini-batched algorithm used to learned this model.

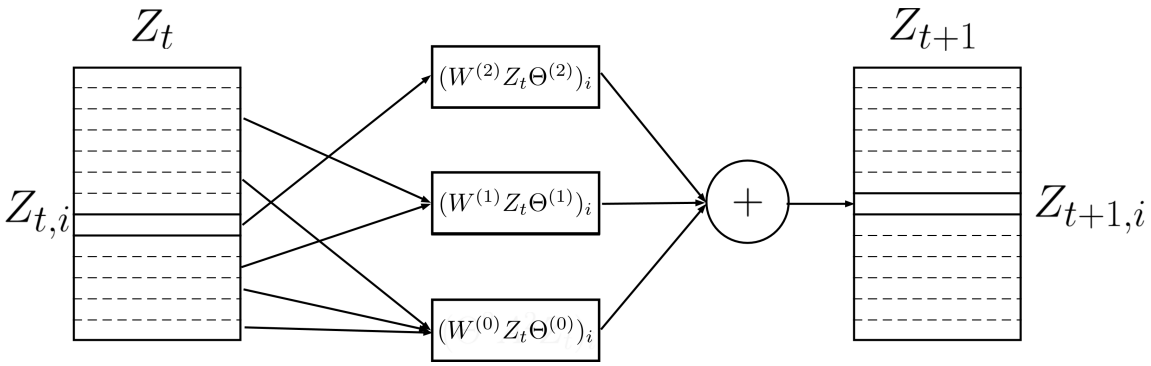


Figure 3.1 – Architecture of the STNN model as described in Section 3.2.4

.

3.2.4 Modeling Different Types of Relations

The model in Section 3.2.3 considers that all the spatial relations are of the same type (e.g. based on sources proximity). For many problems, we have to consider different types of relations. For instance, when sensors correspond to physical locations and the target is some meteorological variable, the relative orientation or position of two sources may imply a different type of dependency between the sources. The multi-relational framework generalizes the previous formulation of the model and allows us to incorporate more abstract relations, like different measures of proximity or similarity between sources. For instance, when sources are spatially organized in a graph, it is possible to define graph kernels, each one of them modeling a specific similarity. The following multi-relational formulation is based on adjacency matrices, and can directly incorporate such graph kernels.

Each possible relation type is denoted r and is associated with a matrix $\boldsymbol{W}^{(r)} \in \mathbb{R}_+^{n \times n}$. For now, and as before, we consider that the $\boldsymbol{W}^{(r)}$ matrices are provided as prior knowledge. Each type of relation r is associated with a transition matrix $\Theta^{(r)}$. This learned matrix captures the spatio-temporal relationship between the series for this particular type of relation. The model dynamics writes:

$$\boldsymbol{Z}_{t+1} = h(\boldsymbol{Z}_t \Theta^{(0)} + \sum_{r \in \mathcal{R}} \boldsymbol{W}^{(r)} \boldsymbol{Z}_t \Theta^{(r)}), \tag{3.4}$$

where $\mathcal{R}$ is the set of all possible types of relations. The learning problem is similar to Equation 3.3 with the argument of h replaced by the expression in Equation 3.4. The corresponding model is illustrated in Figure 3.1. This dynamic model aggregates the latent representations of the series for each type of relation and then applies $\Theta^{(r)}$ on this aggregate. Each $\Theta^{(r)}$ is able to capture the dynamics specific to relation (r).

3.3 Capturing Spatio-Temporal Correlations

In the previous sections, we made the hypothesis that the spatial relational structure and the strength of influence between series were provided as prior information to the model through the $\boldsymbol{W}^{(r)}$ matrices. We introduce below an extension of the model where weights on these relations are learned. This model is denoted STNN-R. We further show that this model can be easily extended to learn both the relations and their weights directly from the data, without any prior knowledge on the spatial structures. This extension is denoted STNN-D.

We first introduce the STNN-R extension. Let $\Gamma^{(r)} \in \mathbb{R}^{n \times n}$ be a matrix of weights such that $\Gamma^{(r)}_{i,j}$ is the strength of the relation between series i and j in the relation r. Let us extend the formulation in Equation 3.4 as follows:

$$\boldsymbol{Z}_{t+1} = h(\boldsymbol{Z}_t \Theta^{(0)} + \sum_{r \in \mathcal{R}} (\boldsymbol{W}^{(r)} \odot \Gamma^{(r)}) \boldsymbol{Z}_t \Theta^{(r)}), \tag{3.5}$$

where $\Gamma^{(r)}$ is a matrix to be learned, $\boldsymbol{W}^{(r)}$ is a prior i.e a set of observed relations, and $\odot$ is the element-wise multiplication between two matrices. The learning problem can now be written as:

$$\begin{aligned} d^*, \boldsymbol{Z}^*, \Theta^*, \Gamma^* = \arg\min_{d, \boldsymbol{Z}, \Gamma} & \frac{1}{T} \sum_t \Delta(d(\boldsymbol{Z}_t), \boldsymbol{X}_t) + \gamma |\Gamma| \\ & + \lambda \frac{1}{T} \sum_{t=1}^{T-1} ||Z_{t+1} - h(\sum_{r \in (R)} (\boldsymbol{W}^{(r)} \odot \Gamma^{(r)}) \boldsymbol{Z}_t \Theta^{(r)})||^2, \end{aligned}$$

where $|\Gamma^{(r)}|$ is a l_1 regularizing term that aims at sparsifying $\Gamma^{(r)}$. We thus add a hyper-parameter γ to tune this regularization factor.

If no prior information is available, then simply removing the $\boldsymbol{W}^{(r)}s$ from equation Equation 3.5 leads to the following STNN-D model:

$$\boldsymbol{Z}_{t+1} = h(\boldsymbol{Z}_t \Theta^{(0)} + \sum_{r \in \mathcal{R}} \Gamma^{(r)} \boldsymbol{Z}_t \Theta^{(r)}),$$

where $\Gamma^{(r)}$ is no more constrained by $\boldsymbol{W}^{(r)}$ so that it will represent both the relational structure and the relation weights. Both models are learned with SGD, in the same way as described in Section 3.2.2. The only difference is that a gradient step on the $\Gamma^{(r)}s$ is added.

3.4 STNN for Data Imputation

We investigate here how the STNN model can be adapted for the data imputation problem.

In the formulation of the data imputation task, in addition to the series values $\boldsymbol{X} \in \mathbb{R}^{T \times n \times m}$ (with T the number of timesteps, n the number of series, and m the dimensionality of the series), we also consider a missing data mask $\boldsymbol{M} \in \{0,1\}^{T \times n}$. M_t^i is the binary mask on the series i at timestep t, and is equal to 1 when vector $X_t^i \in \mathbb{R}^m$ is missing, and 0 if it is present. The goal is to minimize the prediction error of missing data points.

As opposed to the forecasting task, we suppose that observations from every timestep (past and future) are present, and missing data may appear at any timestep. If X_t^i is a missing value for series i at time t, the prediction $\hat{X}_t^i$ is computed based on all the available $X_{t'}^j$, for $j \in \{1, n\}$ and $t' \in \{1, T\}$.

The training objective for the imputation task writes:

$$\mathcal{L}(d, g, \boldsymbol{Z}) = \frac{1}{\sum_t \sum_i (1 - M_t^i)} \sum_t \sum_i (1 - M_t^i) \Delta(d(Z_t^i), X_t^i) + \sum_{t=1}^{T-1} ||\boldsymbol{Z}_{t+1} - g(\boldsymbol{Z}_t)||^2.$$

In this expression, supervision comes from the available X_t^i, so that the Z_t^i value inferred for a missing X_t^i depends on all available observations $X_t'^j$. The second term $\sum_{t=1}^{T-1} ||\boldsymbol{Z}_{t+1} - g(\boldsymbol{Z}_t)||^2$ acts as a regularizer for the Z_t^i value associated to a missing X_t^i. Intuitively, the Z_t^i value for a missing observation should be coherent with the neighboring latent states $\boldsymbol{Z}$ associated to observations $\boldsymbol{X}$. For example, if one supposes that $\boldsymbol{X}_t$ is missing while $\boldsymbol{X}_{t-1}$ and $\boldsymbol{X}_{t+1}$ are observed, the term $\boldsymbol{Z}_t$ is directly constrained by the two loss terms $||g(\boldsymbol{Z}_t) - \boldsymbol{Z}_{t-1}||^2$ and $||\boldsymbol{Z}_t - g(\boldsymbol{Z}_{t-1})||$. In the proposed method, since the model learns one latent state for each series and each timestep, it learns these latent states over missing values that could then be retrieved using the decoding function. Inference in this model is straightforward: once learned, a missing value of series i at timestep t can be computed as $X_t^i = d(Z_t^i)$.

3.5 Experiments

The following section contains experiments and results in different tasks and settings [2]. First, we present experiments on a synthetic dataset to demonstrate some properties of the STNN model and its variants. We, then present results on real-world datasets for the forecasting task, followed by a qualitative analysis of the relation discovery capabilities of our model. Finally, we present results on the data imputation task.

2. Code available at https://github.com/edouardelasalles/stnn

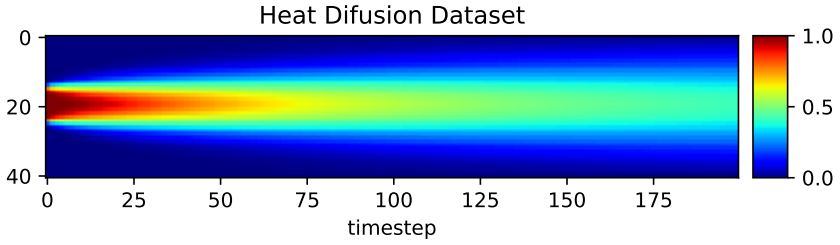


Figure 3.2 – Visualisation of the heat data generated using Equation 3.6 on a segment discretized in 41 points, for 200 timesteps.

3.5.1 Synthetic Experiments on Heat Diffusion

We begin by evaluating and analyzing the STNN model and its variants on a heat diffusion simulation dataset. It is a simple problem whose characteristics, in particular the spatio-temporal dependencies, are perfectly known. Hence, its complexity can be controlled. We consider a 1-D segment where a heat source is applied on its center. The diffusion of the heat is governed by the following differential equation:

$$\frac{\partial u}{\partial t} = a\left(\frac{\partial^2 u}{\partial x^2}\right),$$

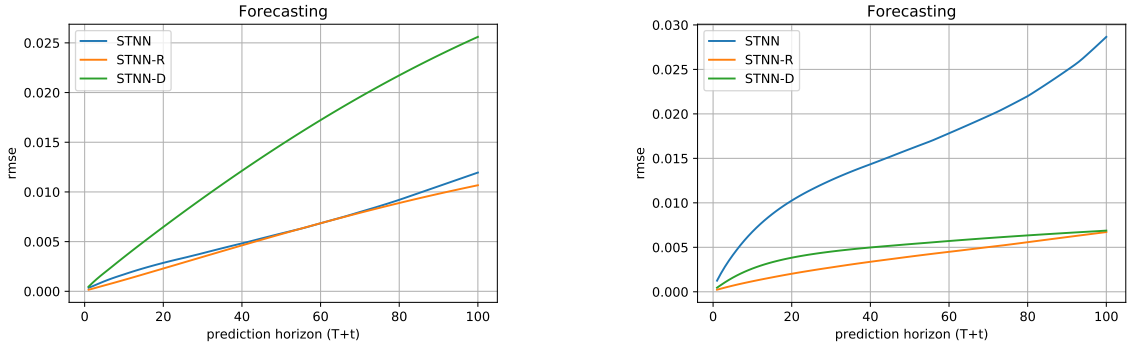
where u is the heat value, a a diffusion constant, and x and t are respectively space and time variables. This equation can be discretized in space and time by the explicit Euler method as follows:

$$u_{t+1}^i = u_t^i + a\Delta t\left(\frac{u_t^{i-1} - 2u_t^i + u_t^{i+1}}{\Delta x^2}\right) \quad \forall i \in 1, ..., n. \tag{3.6}$$

We construct a dataset by simulating heat diffusion on a segment divided into $n = 41$ points, that each corresponds to a series, through 200 timesteps. We use the first 100 timesteps for training and the remaining 100 for the forecasting evaluation. The adjacency matrix $\boldsymbol{W}$ we use for STNN and STNN-R is the one connecting direct neighbors in the diffusion segment. The result dataset can be visualized in Figure 3.2.

Figure 3.3a shows the Rooted Mean Square Error (RMSE) scores for prediction at $t+1$ to $t+100$, and Figure 3.4 shows the predicted values and the ground truth. As expected, the STNN-R model performs best. It has both a strong relational prior (i.e only adjacent points interact with each other) and enough flexibility to adjust the relation weights and well capture the spatio-temporal correlations. STNN-D has no spatial prior, and fails to learn the dynamics of the process.

On the top right image in Figure 3.4, we can see that the errors made by the STNN model are concentrated on the borders: the model over-estimates the heat diffusion at these points. The border points are influenced only by points closer



(a) Heat diffusion experiment

(b) Modulated heat diffusion experiment

Figure 3.3 – Forecasting performances (RMSE) for the synthetic heat diffusion experiments. Left: standard heat diffusion. Right: heat diffusion with modulated diffusion constant. Datasets are simulated for 200 timesteps. Models are learned on the first 100 timesteps and forecast the next 100 timesteps.

to the heat source, whereas other points have a "colder" and a "hotter" neighbor. STNN learns to use both these points in its dynamic function. However, border points are only linked to a single "hotter" point, and hence their heat values are over-estimated. STNN-R, on the other hand, is able to adapt relation weights in order to cope with this side effect. This is illustrated in the right image, second row in Figure 3.4 where the absolute error on the borders is clearly lower for STNN-R than for the other variants.

The relations weights learned by STNN-R (denoted by Γ in Equation 3.5) are shown in Figure 3.5a. A pixel at position (i, j) on this image corresponds to the weight that STNN-R puts on series j at time t for computing the latent representation of series i at time $t+1$. Higher values mean stronger influence of series j in the update of series i. One can see that STNN-R learns asymmetrical and low-value weights between points close to the borders (upper left and bottom right pixels in Figure 3.5a), allowing it to prevent heat from accumulating too quickly at borders. We also show in Figure 3.5b the relation weights discovered by STNN-D. Similar to STNN-R, it learns low relation weights between points near the edges. The horizontal and vertical axial symmetries in Figure 3.5b reflect the symmetry of the dataset itself (Figure 3.4 top left image).

To further explore the adaptivity of STNN, we make the diffusion process more complex. The diffusion constant a is replaced by a Radial Basis Function (RBF) kernel positioned on the center of the diffusion segment:

$$u_{t+1}^i = u_t^i + a^i \Delta t \left(\frac{u_t^{i-1} - 2u_t^i + u_t^{i+1}}{\Delta x^2} \right) \quad s.t.\ a^i = a' K(|i - \frac{n-1}{2}|, \frac{n-1}{2}),$$

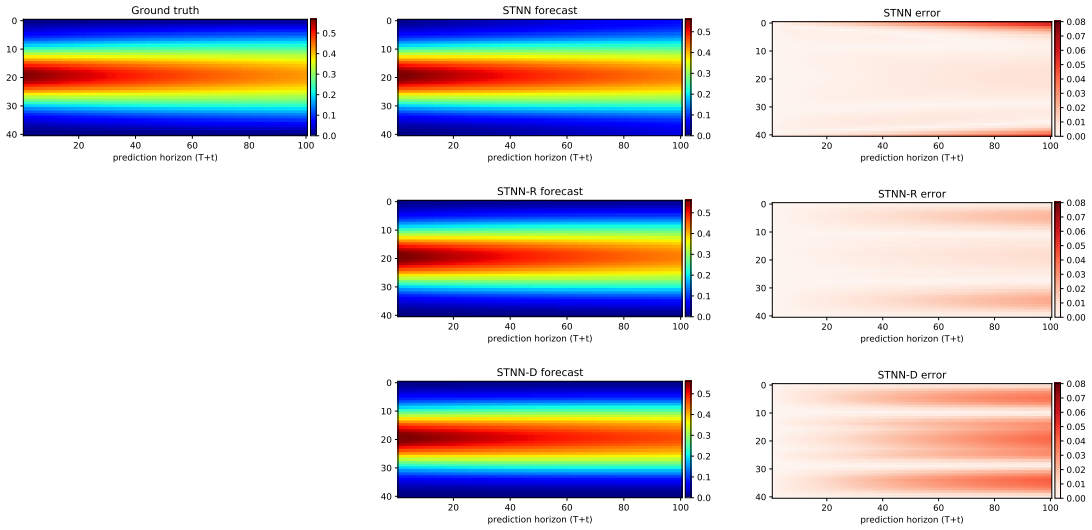
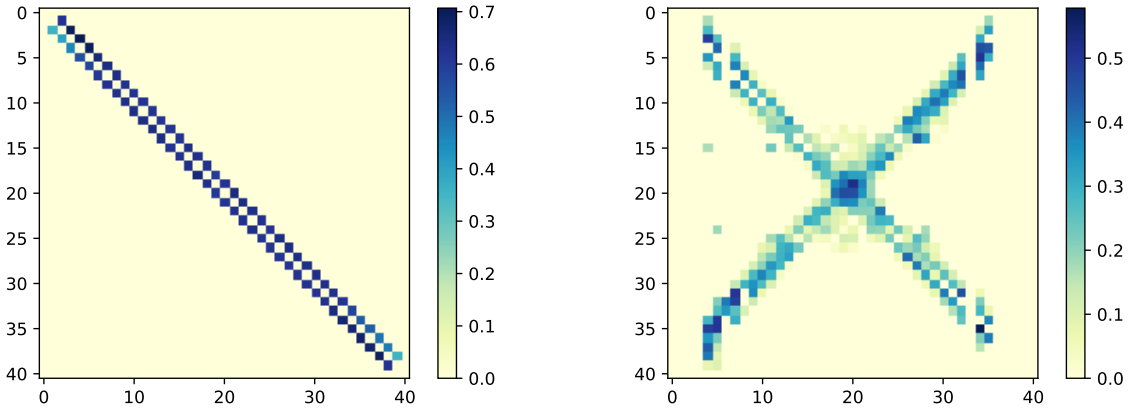


Figure 3.4 – One hundred timestep forecasting of heat diffusion by our three different models.



(a) STNN-R (b) STNN-D

Figure 3.5 – Relation weights learned by STNN-R (left) and STNN-D (right). STNN-R learns the same value for each relation, except on borders where it puts lower weights to prevent over-estimation of heat diffusion due to a border effects in the model.

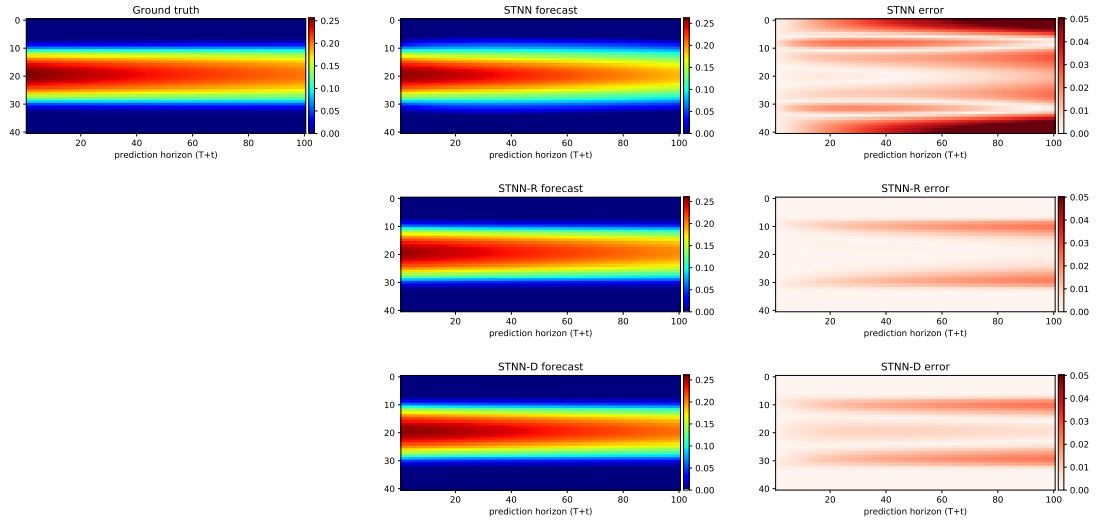
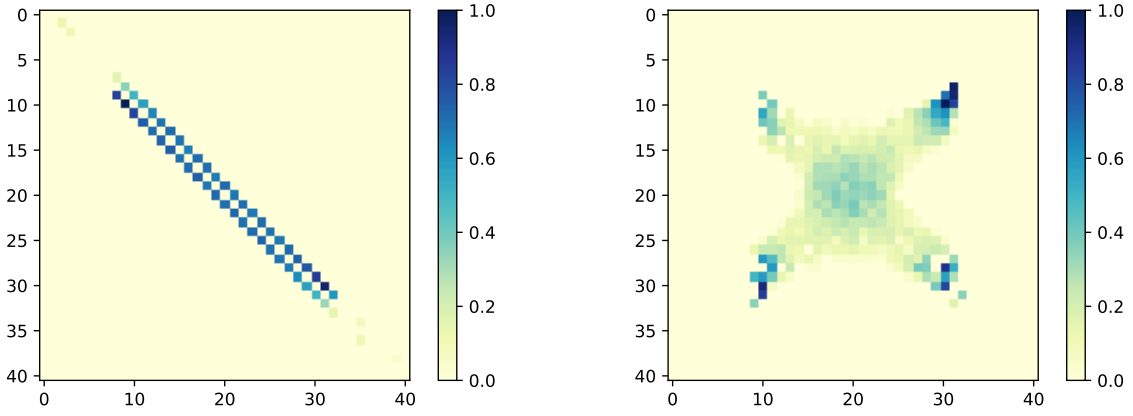


Figure 3.6 – One hundred timestep forecasting of modulated heat diffusion by our three different models.



(a) STNN-R (b) STNN-D

Figure 3.7 – Relation weights learned by STNN-R (left) and STNN-D (right) on modulated heat diffusion. Both model learn to put low weights on borders to match the data. Once again, due to the symmetry of the data, STNN-D learns symmetric weights.

where K is a RBF kernel. This modification results in heat propagating faster in the center, and slower as it reaches the borders.

Results are shown in Figure 3.3b. As expected, the performances of STNN degrade significantly since the prior spatial information provided to the model does not relate to the true process anymore. But thanks to their ability to adjust the relations weights individually, STNN-R and STNN-D maintain good performances. Figure 3.6 shows the predicted values. It is easy to see that STNN over-estimates heat propagation speed. Figure 3.7 shows the learned relations: both STNN-R and STNN-D put very low values on relations between points at the extremities of the segment.

3.5.2 Spatio-Temporal Series Forecasting

For this first task, experiments are performed on a series of spatio-temporal forecasting problems representative of different domains. We consider predictions within a $+5$ horizon i.e. given a training series of size T, the evaluation of the quality of the model will be made over $T+1$ to $T+5$ timesteps. The different model hyper-parameters are selected using a time series cross-validation procedure called rolling origin as in Taieb et al. 2014; Ganeshapillai et al. 2013. This protocol makes use of a sliding window of size T': on a series of length T, a window of size T' is shifted by a constant value k several times in order to create a set of folds. The beginning of each fold is used for training and the remaining for testing. T' is fixed so that it is large enough to capture the main dynamics of the different series. Each series was re-scaled between 0 and 1.

Models and Baselines

We performed experiments with the following models:
(i) **Mean**: a simple heuristic which predicts future values of a series as the mean of its observed past values computed on the T' training steps of each training fold.
(ii) **AR**: a classical univariate Autoregressive (AR) model.
(iii) **VAR-MLP**: a Vectorial AR model where the predicted values of the series at time $t+1$ depend on the past values of all the series for a lag of size R. The predictive model is a MLP with one hidden layer. Its performances were better than a linear Vectorial AR model. Here again the hidden layer size and the lag R were set by cross-validation
(iv) **RNN-tanh**: a vanilla RNN with one hidden layer of recurrent units and tanh non-linearities. Note that this model has the potential to capture the spatial dependencies since all the series are considered simultaneously, but it does not model them explicitly.
(v) **RNN-GRU**: same as the **RNN-tanh**, but the recurrent unit is replaced with a

Table 3.1 – Datasets statistics. n is the number of series, m is the dimension of each series, *timestep* corresponds to the duration of one timestep and *#folds* corresponds to the number of temporal folds used for testing. For each fold, evaluation has been made on the next 5 values at $T+1, T+2, ..., T+5$. The relation columns specify the number of different relation types used in the experiments i.e the number of $\boldsymbol{W}^{(r)}$ matrices used in each dataset.

Dataset	n	m	nb relations	timestep	total length (T)	training length (T')	#folds
Google Flu	29	1	1	weeks	$\approx$ 10 years	2 years	50
GHO (25 datasets)	91	1	1	years	45 years	35 years	5
Wind	500	2	1	hours	30 days	10 days	20
PST	2520	1	8	months	$\approx$ 33 years	10 years	15
Bejing	5000	1	1	15 min	1 week	2 days	20

Gated Recurrent Unit (GRU) [3]. We have experimented with several architectures, but using more than one layer of GRU units did not improve the performance, so we used 1 layer in all the experiments.
(vi) **Dynamic Factor Graph (DFG)**: the model proposed in Mirowski et al. 2009 and presented in Section 2.1.3 is the closest to ours but uses a joint vectorial latent representation for all the series as in the RNNs, and does not explicitly model the spatial relations between series.
(vii) **STNN**: our model, where g is the function described in Equation 3.4, h is the *tanh* function, and d is a linear function. Note that other architectures for d and g have been tested (e.g. MLP) without improving the quality of the prediction. The λ value has been set by cross-validation.
(viii and ix) **STNN-R** and **STNN-D**.

To achieve the best possible results, we grid-searched hyper-parameters for each model and baseline, and for each dataset. Hence, the results presented in this section come from models optimized and fine-tuned independently across all datasets. Each hyper-parameter is selected by cross-validation on a grid of hyper-parameters values.

Datasets

The different forecasting problems and the corresponding datasets are described below. The dataset characteristics are provided in Table 3.1.

- **Disease spread forecasting:** The **Google Flu** dataset contains for 29 countries, about ten years of weekly estimates of influenza activity computed by aggregating Google search queries (see `http://www.google.org/flutrends`).

3. We also performed tests with Long Short-Term Memory (LSTM) and obtained similar results as with GRU.

We extract binary relations between the countries, depending on whether or not they share a border, as a prior $\boldsymbol{W}$.

- **Global Health Observatory (GHO):** This dataset made available by the Global Health Observatory (http://www.who.int/en/) provides the number of deaths for several diseases. We picked 25 diseases corresponding to **25 different datasets**, each one composed of 91 time series corresponding to 91 countries (see Table 3.1). Results are averages over all the datasets. As for Google Flu, we extract binary relations $\boldsymbol{W}$ based on borders between the countries.

- **Geo-Spatial datasets:** The goal is to predict the evolution of geophysical phenomena measured on the surface of the Earth.
  The **Wind** dataset (www.ncdc.noaa.gov/) consists of hourly summaries of meteorological data. We predict wind speed and orientation for approximately 500 land stations on U.S. locations. In this dataset, the relations correspond to a clamped spatial proximity between the series. Given a selected threshold value d, two sources are connected ($w_{i,j} = 1$) if their distance is below d and not connected ($w_{i,j} = 0$) otherwise.
  The **Pacific Sea Temperature (PST)** dataset is a grid (at a 2 by 2 degrees resolution, corresponding to 2520 spatial locations) containing monthly sea surface temperature on the Pacific for 399 consecutive months from January 1970 through March 2003. The goal is to predict future temperatures at different spatial locations. Data were obtained from the Climate Data Library at Columbia University (http://iridl.ldeo.columbia.edu/). Since the series are organized on a 2D grid, we extract 8 different relations: one for each cardinal direction (north, north-west, west, etc...). For instance, the relation *north* is associated with a binary adjacency matrix $\boldsymbol{W}^{(north)}$ such that $W_{i,j}^{(north)}$ is set to 1 if and only if source j is the pixel just above on the satellite image.

- **Car Traffic Forecasting:** The goal is to predict car traffic on a network of streets or roads. We use the **Beijing dataset** presented in Yuan et al. 2010 and Yuan et al. 2011 which consists of GPS trajectories for $\sim$ 10500 taxis during a week, for a total of 17 million points corresponding to road segments in Beijing. From this dataset, we extracted the traffic-volume aggregated on a 15 minutes window for 5000 road segments. The objective is to predict the traffic at each segment. We connect two sources if they correspond to road segments with a shared crossroads.

For all the datasets but PST, we defined the relational structure using a simple adjacency matrix $\boldsymbol{W}$.

Table 3.2 – Average RMSE for the different datasets computed for T+1, T+2,...,T+5. Standard deviation was computed by re-training the models on different seeds.

Models	Google Flu	GHO (averaged)	Beijing	Speed	Direction	PST
MEAN	0.175	0.335	0.201	0.191	0.225	0.258
AR	0.101 ± 0.004	0.299 ± 0.008	0.075 ± 0.003	0.082 ± 0.005	0.098 ± 0.016	0.150 ± 0.002
VAR-MLP	0.095 ± 0.004	0.291 ± 0.004	0.070 ± 0.002	0.071 ± 0.005	0.111 ± 0.140	0.132 ± 0.003
DFG	0.095 ± 0.008	0.288 ± 0.002	0.068 ± 0.005	0.070 ± 0.004	0.092 ± 0.006	0.990 ± 0.019
RNN-tanh	0.082 ± 0.008	0.287 ± 0.011	0.075 ± 0.006	0.064 ± 0.003	0.090 ± 0.005	0.141 ± 0.010
RNN-GRU	0.074 ± 0.007	0.268 ± 0.070	0.074 ± 0.002	0.059 ± 0.009	0.083 ± 0.005	0.104 ± 0.008
STNN	0.066 ± 0.006	$\mathbf{0.261 \pm 0.009}$	0.056 ± 0.003	$\mathbf{0.047 \pm 0.008}$	$\mathbf{0.061 \pm 0.008}$	0.095 ± 0.008
STNN-R	$\mathbf{0.061 \pm 0.008}$	$\mathbf{0.261 \pm 0.010}$	$\mathbf{0.055 \pm 0.004}$	$\mathbf{0.047 \pm 0.008}$	$\mathbf{0.061 \pm 0.008}$	$\mathbf{0.080 \pm 0.014}$
STNN-D	0.073 ± 0.007	0.288 ± 0.090	0.069 ± 0.010	0.059 ± 0.008	0.073 ± 0.008	0.109 ± 0.015

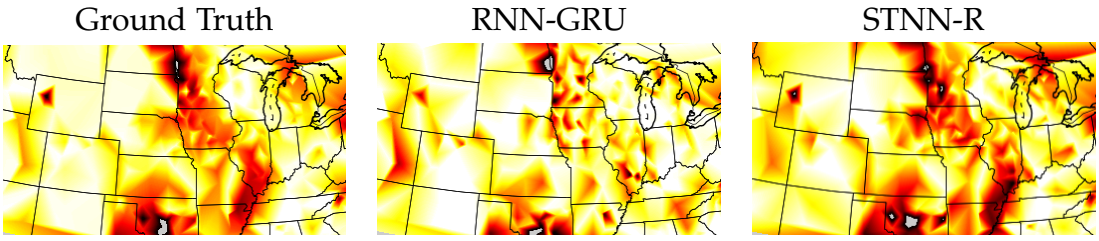


Figure 3.8 – Prediction of wind speed over around 500 stations on the US territory.Prediction is shown at timestep $T + 1$ for RNN-GRU (center) and STNN-R (right).

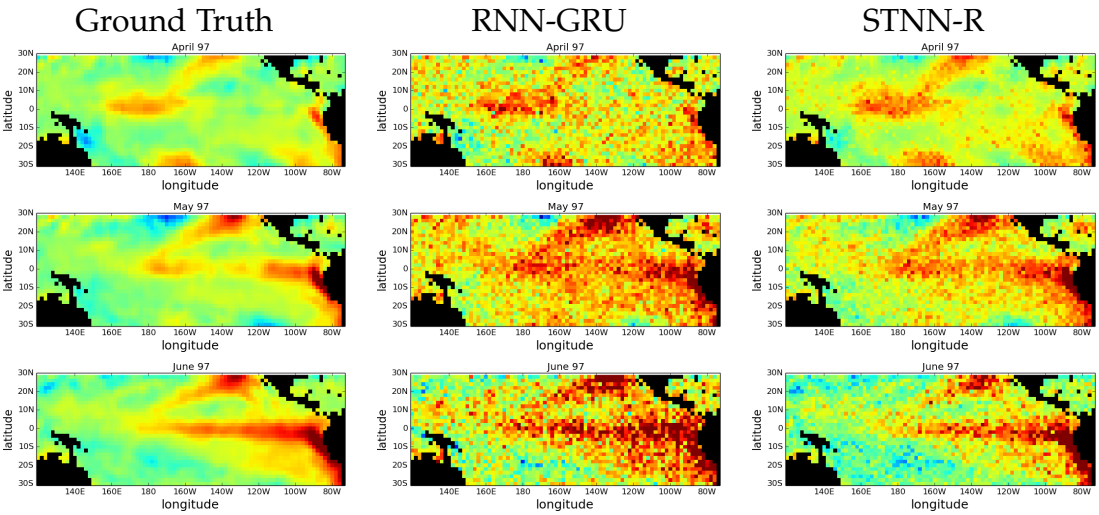
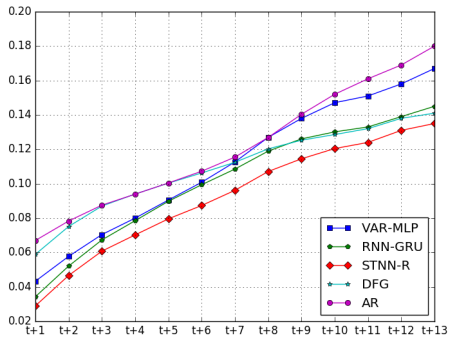
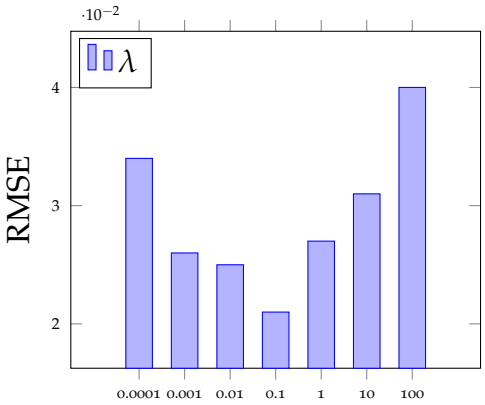


Figure 3.9 – Example of a 3 months prediction of Pacific temperature. The left column is the ground truth and the central and right columns correspond respectively to RNN-GRU and STNN-R predictions at horizon $T + 1$, $T + 2$ and $T + 3$ (top to bottom).



(a) RMSE on the Google Flu dataset at horizon $T+1$ to $T+13$



(b) RMSE on Google Flu w.r.t λ

Figure 3.10 – Quantitative study on the Google Flu dataset.

Results

A quantitative evaluation of the different models and the baselines, on the different datasets, is provided in Table 3.2. All the results are average prediction error for $T+1$ to $T+5$ predictions. The score function used is the Rooted Mean Square Error (RMSE). A first observation is that STNN and STNN-R models, which make use of prior spatial information, significantly outperform all the other models on all the datasets. For example, on the challenging PST dataset, our models increase by 23% the performance of the GRU-RNN baseline. The increase is more important when the number of series is high (geo-spatial and traffic datasets) than when it is small (disease datasets). In these experiments, STNN-D is on par with RNN-GRU or better. The two models do not use prior information on spatial proximity. Vectorial AR logically improves on mono-variable AR (not shown here) and non-linear MLP-VAR improves on linear VAR.

Figure 3.8 and Figure 3.9 illustrate respectively the prediction of STNN-R and RNN-GRU on the meteorology and the oceanography datasets along with the ground truth. Clearly, on these datasets, STNN qualitatively performs much better than RNNs by using explicit spatial information. STNN is able to predict fine details corresponding to local interactions when RNN based models produce a much more noisy prediction. These illustrations are representative of the general behavior of the two models.

We also provide models performances at different prediction horizons $T+1, T+2, \ldots, T+13$ in Figure 3.10a for the Google Flu dataset. Results show that STNN-R performs better than the other approaches for all the prediction horizons and is thus able to better capture longer-term dependencies.

Figure 3.10b illustrates the RMSE of the STNN-R model when predicting at $T+1$ on the Google Flu dataset for different values of λ. One can see that the best performance is obtained for an average value of λ: low values corresponding

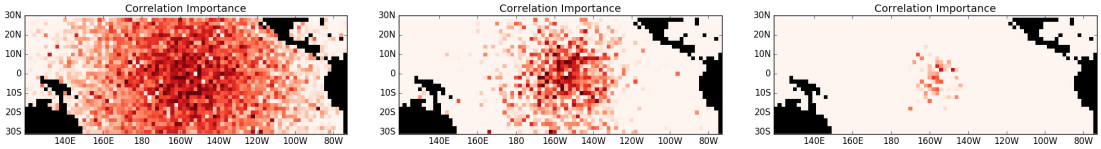


Figure 3.11 – Illustrations of correlations Γ discovered by the STNN-D model with the center pixel as reference, with γ in $\{0.01, 0.1, 1\}$ (from top to bottom). We can see that the value of γ constrains the spatial range of relations learned by STNN-D.

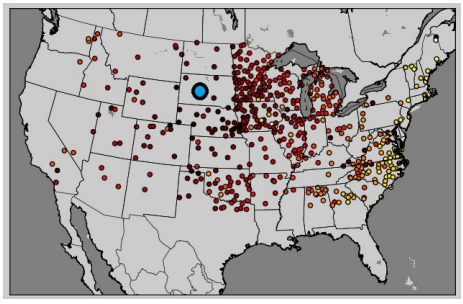


Figure 3.12 – Spatial correlation discovery with STNN-D on the Wind dataset. The blue point is the reference, and the others represent the weight learned by STNN-D with the reference. We can see that points close to the reference are assigner higher weights compared to more distant ones.

to weak temporal constraints do not allow the model to learn the dynamics of the series and lead to overfitting, while high values degrade the performance of STNN since not enough weight is put on the reconstruction loss.

3.5.3 Discovering the Spatial Correlations

In this subsection, we illustrate the ability of STNN to discover relevant spatial correlations on different datasets. Figure 3.11 and Figure 3.12 illustrate the values of Γ obtained by STNN-D where no structure (e.g. adjacency matrix W) is provided to the model on the PST and Wind dataset respectively. Each pixel corresponds to a particular time series and the figure shows the correlation $\Gamma_{i,j}$ discovered between each series j with a series i. The series i is roughly located at the center of the picture in Figure 3.11, and is represented by a blue circle in Figure 3.12. The darker a pixel is, the higher the absolute value of $\Gamma_{i,j}$ is (note that black pixels correspond to countries and not sea). Different levels of sparsity are illustrated from low (up) to high (down). Even if the model does not have any knowledge about the spatial organization of the series (no $\boldsymbol{W}$ matrix provided),

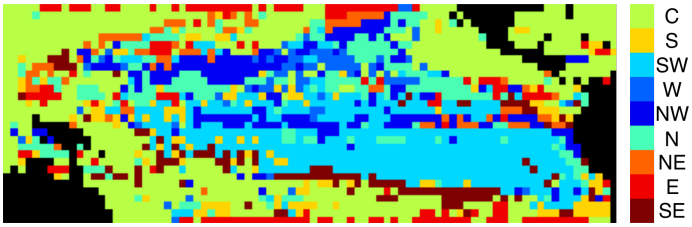


Figure 3.13 – Spatial correlations extracted by the STNN-R model on the PST dataset. The color of each pixel corresponds to the principal relation extracted by the model.

it is able to re-discover this spatial organization by detecting strong correlations between close series, and low ones for distant series.

Figure 3.13 illustrates the correlations discovered on the PST dataset. We used as priors 8 types of relations corresponding to the 8 cardinal directions (South, South-West, etc...). In this case, STNN-R learns weights (i.e $\Gamma^{(r)}$) for each relation based on the prior structure. For each series, we plot the direction with the highest learned weight. The strongest direction for each series is illustrated by a specific color in the figure. For instance, a dark blue pixel indicates that the stronger spatial correlation learned for the corresponding series is the North-West direction. The model extracts automatically relations corresponding to temperature propagation directions in the pacific, providing relevant information about the spatio-temporal dynamics of the system.

The model can be adapted to different situations. Figure 3.14 represents the captured temporal evolution of the spatial relations on the PST dataset. For this experiment, we have slightly changed the STNN-R model by making the $\Gamma^{(r)}$ time-dependent according to:

$$\Gamma^{(r)}_{t,i} = f_r(\boldsymbol{Z}^i_t),$$

with f_r an MLP with a logistic activation function. This means that with this modified model, the spatial relation weights depend on the current latent state of the corresponding series and may evolve with time. This allows use to predict dynamic relations. On Figure 3.14, the different plots correspond to successive timesteps. The color represents the actual sea surface temperatures, and the arrows represent the direction of the stronger relation weights $\Gamma^{(r)}_t$ among the eight possible directions (N, NE, etc). One can see that the model captures coherent dynamic spatial correlations such as global currents directions or rotating motions that gradually evolve with time.

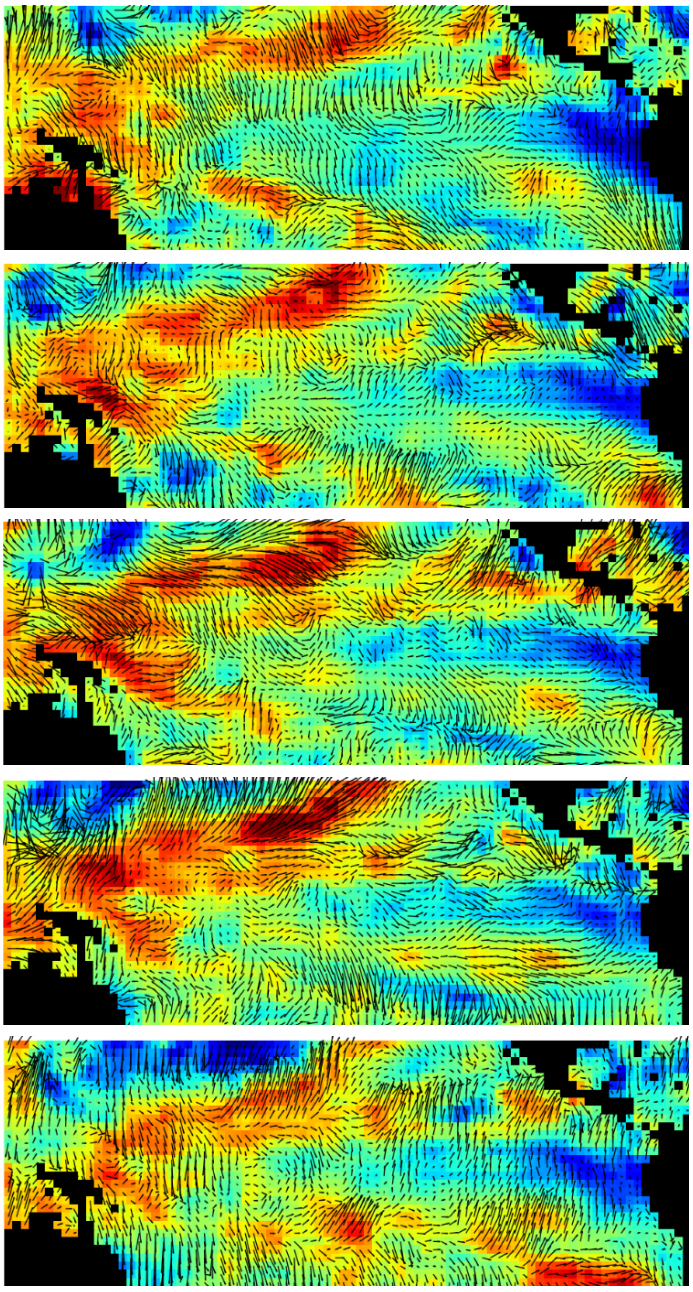


Figure 3.14 – Dynamic spatio-temporal relations extracted from the PST dataset on the training set on 3 consecutive timesteps. Colors represent the actual sea surface temperature. Arrows represent the extracted spatial relations that evolve through time.

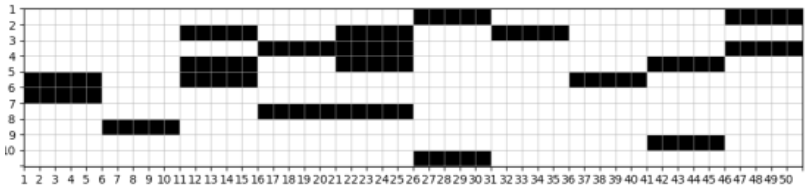


Figure 3.15 – The figure represents 10 time series over 50 timesteps, white squares corresponding to observed values and black squares corresponding to missing ones. These missing values have been generated from a fully observed set of time series using a corruption schema where $p_m = 0.2$ and $l_m = 5$ (see Figure 3.5.4).

3.5.4 Data Imputation

This section presents the experiments concerning data imputation. We first introduce the experimental protocol, we briefly describe the baselines and then detail and comment on our quantitative and qualitative results.

Experimental Protocol

As mentioned in section 3.4, we focus on the case where the information for some or all of the series is missing at different timesteps. This setting is quite general and covers different situations. For instance, missing values may affect only some of the series at a given timestep, or all the series may be affected at the same timesteps. Besides, the number of timesteps with missing values may be extremely different from a problem to the other. For evaluating the models, one needs a generic protocol. To provide a quantitative evaluation of the quality of our model, we defined a protocol common to all the datasets. For a given dataset, we remove a random subset of the data as detailed below.

We choose a missing rate p_m - different values are used in the experiments - which indicates the proportion of the series values that are going to be considered as missing. For instance, $p_m = 0.2$ means that 20% of the dataset values are considered as missing. We also choose a missing value length l_m, which determines the size of the missing chunks. For instance, if $l_m = 5$, a missing data chunk is composed of 5 consecutive timesteps in a given series. Figure 3.15 shows a sample of a missing data mask $\boldsymbol{M}$ for 50 timesteps with 10 series, where $p_m = 0.2$ and $l_m = 5$.

The training set contains all available observations (non-missing values - i.e. white squares in Figure 3.15) while the test set contains the missing values (black squares in Figure 3.15). In order to select hyper-parameters, we held out a validation set from the training set. More specifically, during the validation phase, we take out a proportion p_m of the training set that we keep for evaluating hyper-parameters, and train on the remaining $1 - p_m$ portion of the training set. Once

Table 3.3 – RMSE for the imputation task on the different datasets. These results where obtained for $p_m = 0.1$ and $l_m = 5$. (X means that the dataset is too large for the available implementation)

Models	Google Flu	Beijing	Wind	PST
MEAN	1.08e−1	8.37e−2	2.28e−1	6.14e−2
LAST	6.96e−2	6.94e−2	1.51e−1	9.86e−2
Amelia II	7.98e−2	6.99e−2	1.87e−1	X
GRU	3.77e−2	5.25e−2	1.32e−1	1.04e−2
DFG	4.04e−2	5.26e−2	1.37e−1	7.77e−3
STNN	3.65e−2	4.85e−2	1.17e−1	**2.59e−3**
STNN-R	**3.20e−2**	**4.52e−2**	1.21e−1	2.76e−3
STNN-D	3.31e−2	4.60e−2	**1.15e−1**	3.78e−3

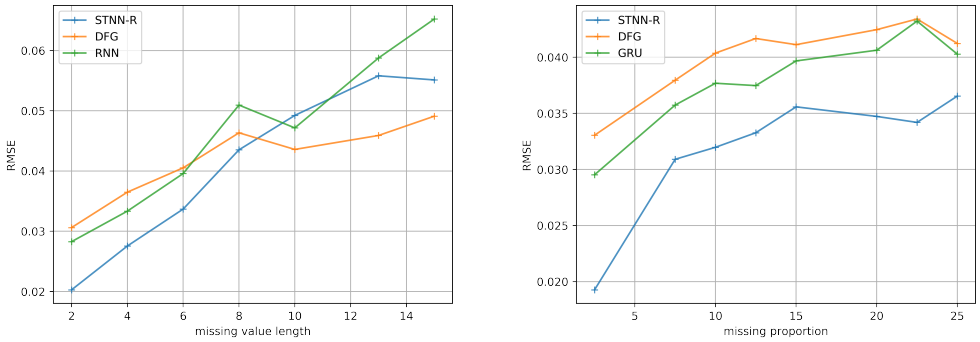
the hyper-parameters are selected, we train the model from scratch on the entire training set and evaluate on the test set.

We evaluate our model on the following datasets GFlu, Wind, Beijing car traffic, and PST. For the Wind dataset, we jointly consider the speed characteristic and the direction characteristic in order to evaluate our model on a multi-variate setting.

Baselines

We compared our model to the following baselines:

- **Mean:** missing data are imputed with corresponding series' average value.
- **Last:** missing data are imputed with the series' last observed value.
- **Amelia II (Honaker et al. 2011) :** a statistical model for missing data imputation based on a bootstrapped version of the Expectation Maximization (EM) algorithm. For our experiments, we sample m values from the model, given the observed variables, and take the mean of these samples.
- **GRU:** we used the "GRU-simple" baseline proposed by Che et al. 2016: for a time series $\boldsymbol{X}$, each missing value is replaced with the average value of the series, giving a new series $\tilde{\boldsymbol{X}}$. At each timestep t, the GRU is fed with the concatenation of $\tilde{\boldsymbol{X}}_t$ and $\boldsymbol{M}_t$, the missing value mask at timestep t. The loss is a standard Mean Square Error (MSE), where the gradient for missing values is not backpropagated.
- **DFG:** the DFG model Mirowski et al. 2009 also used in the prediction experiments, and presented in Section 2.1.3.



(a) Corruption length variation (b) Corruption proportion variation

Figure 3.16 – Evolution of our model and baselines score when the missing value proportion and corruption length change. (Left) length of the occulted chunks varies, while the corruption proportion stays at 10%. (Right) missing proportion changes, while the corruption length stays at 5 times-steps.

Note that Mean and Last are frequently used heuristics for handling missing values, GRU-simple and DFG are state-of-the-art latent dynamical models designed for imputation in time series or sequences.

Quantitative Results

Table 3.3 presents the quantitative test results. The scores are the RMSE on the missing test values. These results were obtained with $p_m = 0.1$ and $l_m = 5$. As for the forecasting results, the STNN model and its variants perform consistently better than the baselines. For the Google Flu dataset, STNN-R is 18% better than the strongest baseline (GRU). On the Wind dataset, STNN-D achieves the best results, performing 15% better than DFG. It is on the PST dataset that we obtain the strongest results: STNN performs 3 times better than DFG. Amelia II baseline fails to reach the performance of deep models (STNN, DFG, GRU) by a large margin. This is due to its over-simplistic normal prior and the lack of spatial prior.

We also performed a quantitative study of the model robustness for different levels of missing values by comparing "STNN-R", "DFG" and "GRU-simple". Results are shown in Figure 3.16a where l_m is varying with a fixed $p_m = 0.1$, and in Figure 3.16b where p_m is varying with a fixed $l_m = 5$. In Figure 3.16a one can see that STNN-R performs better than the two baselines for all missing value proportions (Figure 3.16b). Concerning the missing values length, for lengths higher than 10 timesteps, DFG gets better results than STNN-R. Our model learns one explicit latent factor per time series, and when too many consecutive values are missing for one time series, the predicted latent factors tend to collapse. The DFG model only learns one factor common to all the series and is then more

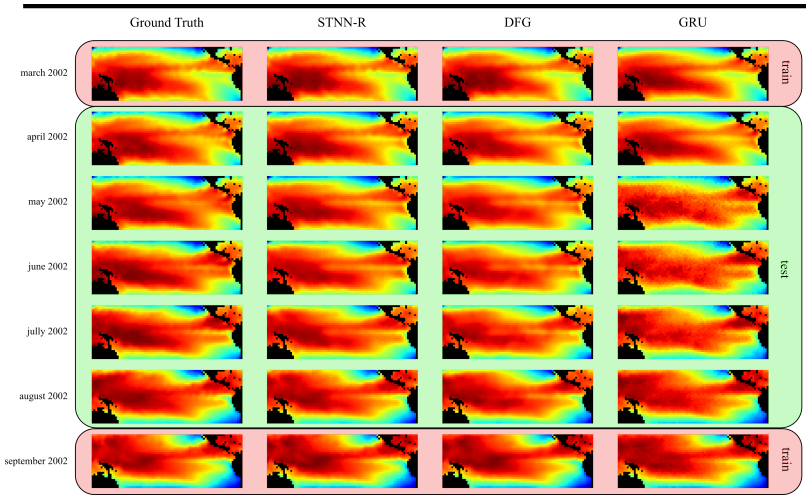


Figure 3.17 – Complete timestep imputation visualization, where all values in the test timesteps where missing during training. August and September 2002, shown with a pink border, are observed and used for training. April to August 2002 included, shown with a green border, are not observed during training and are used as a test set for imputation.

robust to this type of corruption: since in our setting, the missing values only consider a subset of the series, DFG benefits from the other observed values at a given timestep to better infer a correct latent factor.

Complete timestep Reconstruction

We also experiment with a configuration where, at a given timestep, the values of all the series are missing simultaneously. This is a scenario that can happen, for instance, on earth observation problems. For these experiments, we keep a missing value ratio at 10% and a missing sequence length of 5 timesteps, with all the values in any chunks of 5 timesteps completely occulted during training.

Figure 3.17 shows a sample of the data reconstructed by our models and baselines on the Pacific surface temperature dataset from March to September 2002. March and September 2002 where observed (pink border on the figure) and used for training, while observations from April to August 2002 where occulted (green border on the figure) and used in the test set as missing values. Figure 3.17 shows that the GRU baseline performs worse than both STNN-R and DFG: the predictions are not locally smooth. On this figure, the predictions from STNN-R and DFG look very similar. In order to analyze better the differences, we also plot the absolute error performed by the three models in Figure 3.18. The larger error of GRU is again clearly visible on this figure, and it also clearly appears that STNN-R imputations are of better quality than those of DFG which does not model explicitly the spatial dependencies.

Table 3.4 – Test results (RMSE) on the PST dataset with $p_m = 0.1$ and $l_m = 5$, where all values of corrupted timestep are not seen during training. These results come from the same experiment that yields the images in Figure 3.17.

	STNN-R	DFG	RNN
PST	**2.21e−2**	2.83e−2	3.25e−2

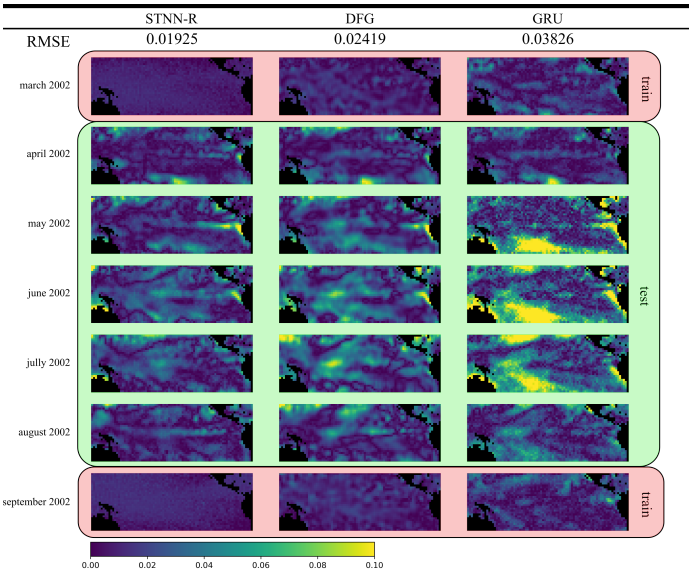


Figure 3.18 – Absolute error visualization for imputed data of Figure 3.17 i.e absolute difference between reconstructed values and ground truth values. The RMSE line corresponds to the RMSE computed only on the 5 test timesteps (green background) indicated in the figure.

The RMSE printed above each column in Figure 3.18 is the RMSE of the 5 reconstructed timesteps for each model. This confirms the visual results: STNN-R actually achieves a better performance on these 5 timesteps. We show in Table 3.4 the RMSE for all the dataset. We can see that STNN-R performs better on average for all the missing chunks.

3.6 Conclusion

In this chapter, we introduced a new DL model for addressing multivariate spatio-temporal time series modeling problems, with applications to forecasting and imputation. We show that DL methods generally surpass existing statistical models on several benchmarks. Our principal contribution was to explicitly model spatial relations between series. We saw that this approach led to better prediction

results: the model achieve good performances on a 5 timesteps forecasting horizon, but predictions tend to converge to the mean series on longer horizons. This is in part due to the stochasticity of the data, that is not modeled in the current framework. Hence, handling this stochasticity is a key challenge in order to build better prediction systems, that we will investigate in the two next chapters of this thesis. Another limitation of the presented model is that the latent states are inferred through gradient descent. This means that nearly all the training procedure has to be performed again for inferring latent states on a new location, for instance. In Chapter 5, we will present a prediction model where the latent state inference procedure is amortized across different series with a Neural Network (NN).

CHAPTER 4

LEARNING DYNAMIC LANGUAGE MODELS AND AUTHOR REPRESENTATIONS

Contents

Chapter abstract

In this chapter, we study the evolution of data generation processes through time. In the previous chapter, we limited ourselves to deterministic modeling. Here, we consider that data is generated at each timestep by a generation process that evolves through time. We propose to learn such a dynamic generative process on textual data. Text is of particular interest for this problem for two reasons: 1) language models are at the heart of numerous works, notably in the text mining and information retrieval communities; 2) language evolves over time with trends and shifts in technological, political, or cultural contexts. Temporal language modeling thus appears as a good subject to study the temporal evolution of data generation processes. We propose to tackle this problematic by augmenting a neural language model with its temporal and author contexts. We first present a temporal model where a global latent variable is structured in time by a learned non-linear transition function. We then integrate authors into the model to capture

language diffusion tendencies in author communities through time. Here, we learn authors and temporal vector states that are able to leverage the latent dependencies between the text contexts.

- Edouard Delasalles, Sylvain Lamprier, and Ludovic Denoyer (2019a). "Dynamic Neural Language Models". In: *Neural Information Processing - 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12-15, 2019, Proceedings, Part III*, pp. 282–294.
- Edouard Delasalles, Sylvain Lamprier, and Ludovic Denoyer (2019b). "Learning Dynamic Author Representations with Temporal Language Models". In: *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pp. 120–129.

4.1 Introduction

In the previous chapter, we proposed a spatio-temporal neural model that produces point estimates of different physical quantities. In this chapter, we are interested in learning dynamic generation models, whose generation process adapts in time. To do so, we propose to study a specific category of generative models: language models for textual data.

We are interested in textual data because various shifts affect language: the meaning of words can shift, new words appear as other vanish, and yesterday's topics are different from tomorrow's. Moreover, textual documents often come with publication dates, making it relatively easy to construct datasets. Finally, to our knowledge, no proper neural language models taking into account publication date were proposed yet.

Early works on language modeling focused on the unigram multinomial model (F. Song et al. 1999), and recent works are shifting toward neural approaches, with distributed representations of words (Y. Bengio et al. 2003; Mikolov et al. 2010). Research on these deep Language Models (LMs) is very active (Vaswani et al. 2017; Merity et al. 2018b; Bai et al. 2018; Melis et al. 2018; Merity et al. 2018a), with applications in various text-related tasks such as speech recognition (Chiu et al. 2018), image captioning (Vinyals et al. 2017), or text generation (Fedus et al. 2018). And more recently, this task gained even more interest in the Natural Language Processing (NLP) community, as a mean to pre-train large multi-task networks (Devlin et al. 2019; Howard et al. 2018; Peters et al. 2018).

To handle temporal evolution in written language, recent research mainly focuses on learning distinct word embeddings per timestep (Hamilton et al. 2016; Kim et al. 2014; Kulkarni et al. 2015) and smoothing them in time (Bamler et al. 2017; Yao et al. 2018; Montariol et al. 2019). Word embeddings are powerful tools to capture and analyze semantic relations between word pairs (Mikolov et al. 2013). However, learning different embeddings for each timestep leads to

learning algorithms with high time and memory complexity, leading to several approximations. For instance, Yao et al. 2018 use alternate optimization that breaks the flow of gradient through time. The smoothing skip-gram approach from Bamler et al. 2017 requires complex gradient estimations, that involve solving tridiagonal linear systems which cannot be parallelized in time.

Moreover, very few works focus on the combined consideration of the writer and the publication date of textual documents. It is in the domain of information diffusion, which studies content transmissions in information networks (Saito et al. 2009), that most of the work on dynamic extraction and prediction of relationships between authors through time has been proposed. However, almost all of the proposed approaches focus on the study of the information spread in a binary setting (infection or non-infection by a content emitted from one source in the network). Now, it appears obvious that dynamics in author communities (inter-author influences or patterns of reactions to some external stimuli) are not limited to binary events, but are also reflected in more diffuse behaviors, and notably on the way people communicate. Various works on topic modeling and their temporal evolution exist (X. Wang et al. 2006; Kabán et al. 2002), but they do not consider the multi-authors setting. Moreover, they are built on bag-of-words representation, and thus cannot directly leverage the representation learning power of deep LMs.

In this chapter, we study language evolution from a deep LM perspective. The aim is to capture the language evolution through time via an end-to-end framework, where a standard Recurrent Neural Network (RNN) is conditioned by a latent representation of temporal drifts in language and/or authors. Incorporating latent random variables in RNNs has already done for textual data (Q. V. Le et al. 2014; Serban et al. 2017; Zaheer et al. 2017a). However, no RNN LMs methods have been proposed for the extraction of temporal or structural dynamics in language and author communities. We first propose a state-based dynamic neural LM that learns transitions between global states through time, rather than focusing on distinct word embeddings. We then study language evolution dynamic in author communities and propose a representation learning model of authors through time. In both cases, we condition a deep LM with state vectors. We conducted experiments on a scientific publications corpus, a news corpus, and a social network corpus for several temporal tasks: modeling (all timesteps are visible), imputation (random timesteps are hidden), and prediction (future timesteps are hidden). Our methods consistently achieve state-of-the-art performances on all tasks. Moreover, we performed quantitative and qualitative studies of the learned latent representations and show that our model is able to learn meaningful representations.

4.2 History of Temporal Language Modeling

Language evolution was tackled more than fifteen years ago through the task of topics evolution in textual documents. Notably, Kabán et al. 2002, with a model based on Hidden Markov Models (HMMs), seek to visualize temporal evolution in a textual stream. This approach falls in the general field of Topic Detection and Tracking, where the idea is to identify and follow trending topics in streams. The approach, which extends the temporal generative topographic mapping of Bishop et al. 1997 for textual modeling, allows one to visualize the thematic changes via trajectories on a two-dimensional grid. However, this kind of work enables tracking of thematics text segmentation, but cannot be used for language modeling. The non-markovian approach proposed in X. Wang et al. 2006 is restricted to bag-of-words representations but has a good ability to detect the topics' evolution over the observation period. Besides, various works studied temporal vocabulary evolution - according to semantic graph transformations in Kenter et al. 2015 -, or thematic shifts in author communities - according to the dominant topics per timestep in Hall et al. 2008.

Closer to applications targeted in this paper, dynamic topic models (Blei et al. 2006) propose an Latent Dirichlet Allocation (LDA)-like modeling (Blei et al. 2003), where the topic distributions and the distributions of words with respect to topics evolve over time. The evolution between successive multinomial distributions are driven by Brownian motions of their natural parameters, in a Kalman filters fashion, and optimized via variational inference. However, these approaches require manually setting the number of topics, and LMs are limited to simple word occurrence distributions. It is not trivial to include models with long-term dependencies, such as Long Short-Term Memories (LSTMs), in this context. Moreover, contrary to ours, these approaches are usually constrained to specific conjugate distributions for the inference of the latent variables of their evolution model. Note the extensions of Blei et al. 2006 to a multi-scale temporal version (Iwata et al. 2012) or a model with continuous-time dependencies (C. Wang et al. 2012). Besides, Gerrish et al. 2010 introduce the concept of influence between documents, which could get closer to our objective but which is limited to analysis tasks. Lastly, E. Wang et al. 2011 propose a temporal approach that considers relationships between documents via a known graph of dependencies, which leaves the scope of this study where we assume that such relational knowledge is not available a priori.

After the introduction of the Word2Vec model (Mikolov et al. 2013), numerous papers proposed derivations of the famous skip-gram algorithm for time annotated corpora (Frermann et al. 2016). All these approaches attempt to acquire a better understanding of language evolution by studying shifts in words semantic through time. Among them, Eger et al. 2016 learn linear temporal dependencies between word representations. Yao et al. 2018 learn diachronic

word representations by matrix factorization with temporal alignment constraints. Bamler et al. 2017 proposed a temporal probabilistic skip-gram model with a diffusion prior. Rudolph et al. 2017a also propose a probabilistic framework that uses exponential embeddings. Compared to HMM and LDA based approaches, the skip-gram algorithm uses standard gradient descent and can be parallelized easily to scale to massive corpora. The goal of these works is to learn some semantic representations of words that can be used directly in various neural models. The temporal dependencies are defined on word representations: each considered timestep is associated with its own vocabulary representation forced to respect various temporal constraints.

However, all these temporal word embedding approaches suffer from a major drawback: complete sets of embeddings must be learned for each timestep. This leads to learning algorithms with high time and memory complexity, requiring several approximations, like alternate optimization that breaks gradient flow through time in Yao et al. 2018, or gradient approximations in Bamler et al. 2017. A notable exception is Rosenfeld et al. 2018 which combine a static word representation to a scalar timestep in a deep neural network that produces a temporal embedding. It appears difficult to consider such kind of approach in a multi-author setting, for which separated representations should be learned both per timestep and per author. We can note the approach of Rudolph et al. 2017b for grouped data, that proposed to reduce the number of parameters by sharing context vectors between groups, but whose transposition to a multi-author setting appears difficult (very high number of groups, doubled dependencies, temporal evolution vs connected groups). Another limitation with this kind of approach is that they do not allow end-to-end learning of LMs, and extending them for outputting word probabilistic distributions is usually difficult.

An alternative to these various models is to leverage RNNs for language modeling. Compared to the skip-gram algorithm that uses a limited context window, recurrent LMs operate on sequences of arbitrary length and can capture long-term dependencies.

4.3 Preliminaries

This chapter is composed of two parts. In Section 4.4, we present a variational recurrent language model with global latent variables for the temporal language modeling task. In Section 4.5, we propose to learn dynamic author representations with a recurrent language model.

For the two parts, we use the same notations described as follows. We consider a corpus $\mathcal{D}$ of N text publications defined over a vocabulary of size V. Each publication $\boldsymbol{x}$ is associated with a publication timestep $t \in \{1, \dots, T\}$ and an

author $a \in A$. A publication is a sequence of tokens $\boldsymbol{x} = \{x_1, x_2, \ldots, x_{|\boldsymbol{x}|}\}$. In both parts, our objective is to propose a recurrent language model of the form:

$$P(\boldsymbol{x}|a,t) = \prod_{k=0}^{|\boldsymbol{x}|} P(x_{k+1}|\boldsymbol{x}_{0:k}, a, t),$$

where x_0 is a special token to indicate the beginning of the document. Note that authors are not taken into account in the first part.

All models and baselines are evaluated on the same three corpora:

- The Semantic Scholar (S2) (Ammar et al. 2018) corpus is composed of titles from scientific papers published in machine learning conferences and journals from 1985 to 2017, split by year (33 timesteps). We lower-cased the texts and used the same WordPiece model as in Devlin et al. 2019 to tokenize the corpus, which has around 30K tokens. The corpus is composed of 45K titles, representing a total of 800K tokens with 1000 authors. The number of titles is not uniformly distributed, and grows quasi-exponentially with time: the year 1985 contains around 100 documents while the year 2017 has around 5K.

- The New York Times (NYT) (Yao et al. 2018) corpus is composed of headlines from the New York Times newspaper spanning from 1990 to 2015, also split by years (26 timesteps). We also lower-cased the texts, but we use the NLTK (Bird 2006) word tokenizer and replaced every number with a special `N` token. Words appearing less than 5 times in the training set were discarded, giving a vocabulary of around 6K tokens. The corpus contains 40K documents, 470K tokens, and 500 authors. In this corpus, the documents are evenly distributed in time.

- The Reddit corpus contains a sample of 3% of the social network's posts presented in Tan et al. 2015. It is composed of 100K posts sampled from January 2006 to December 2013 split by quarters (32 timesteps). Words appearing less than 5 times in the training set were discarded, giving a vocabulary of around 13K tokens.

All models and baselines in this chapter have the recurrent language model backbone. It is 2 layers AWD-LSTM (Merity et al. 2018b) with hidden units and word embeddings of size 400. We use weight dropout, variational dropout, embedding dropout. We also tie word embeddings and decoder weights (Inan et al. 2017). We use the Adam optimizer (Kingma et al. 2015) with mini-batches of size 64, a learning rate of 0.003, and default parameters. Hyper-parameters were tuned by grid search on a dedicated validation set.

4.4 Dynamic Recurrent Language Model

In this part, we propose a dynamic recurrent neural network for language modeling in time annotated document corpora. The model is an State Space Model (SSM) with one global latent state per timestep used to condition an LSTM Language Model. Unlike most current methods that learn complete word embedding matrices for each timestep, we only learn one embedding per word which is augmented with a state of the SSM. The LSTM captures general language dynamic and uses the temporal states to module its dynamics depending on language bias specific to each timestep. We also learn a transition function between states that enables the prediction of future states.

4.4.1 Model

Our goal is to extend classic recurrent language models with a dynamic component to adapt it to language shifts through time. To that aim, we condition an LSTM language model with temporal latent variables. We learn global latent variables structured in time with a transition function learned jointly with the LSTM. The latent variables are global because documents published at the same timestep all share the same latent variable. This allows the LSTM to capture language structures common to the entire dataset, while global latent variables are able to factorize language elements specific to their timestep. A schematic overview of the model is presented in Figure 4.1.

Let $\mathbf{z}_t \in \mathbb{R}^{d_z}$ be the latent variable corresponding to timestep t. The sequence probability of a document $\boldsymbol{x}$ published at timestep t is now computed as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|t) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathbf{z}_t) = \prod_{k=0}^{|\boldsymbol{x}|} p_{\boldsymbol{\theta}}(x_{k+1}|\boldsymbol{x}_{0:k}, \mathbf{z}_t).$$

Note that $\mathbf{z}_t$ depends only on the *timestep* at which $\boldsymbol{x}$ has been published, and not specifically on $\boldsymbol{x}$ itself. In our architecture, we concatenate $\mathbf{z}_t$ to the embeddings of each word x_k as we have found it to work best empirically.

The latent states $\mathbf{z}_t$ are Gaussian random variables structured in time via a dynamic component taking the form of a Gaussian model. Its mean is a function g of the previous state and its covariance is a learned diagonal matrix σ^2:

$$\mathbf{z}_{t+1}|\mathbf{z}_t \sim \mathcal{N}(g(\mathbf{z}_t; \boldsymbol{w}), \sigma^2),$$

where $\boldsymbol{w}$ are the parameters of g. Learning a transition model helps to regularize the inferred latent states, and allows us to predict future states. Moreover, it gives us the possibility to estimate future states of the system, where data is not available during training. The prior's mean on the first timestep is a learned

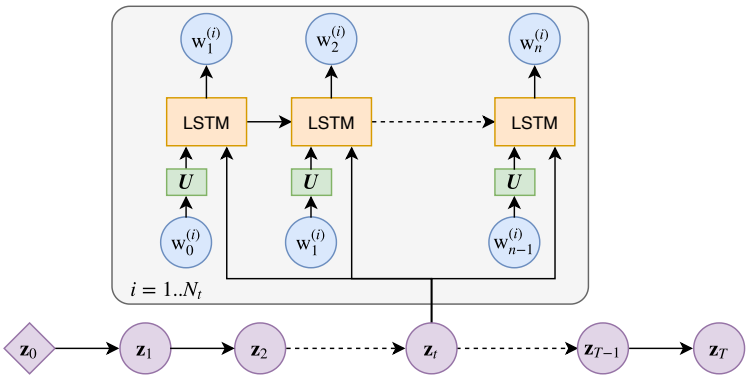


Figure 4.1 – Schematic representation of our dynamic recurrent language model. The temporal variables $\mathbf{z}_t$ are global and are used to condition an LSTM. They are concatenated to the word embeddings (denoted by $\boldsymbol{U}$) of each word in a document.

vector z^0 acting as the initial conditions of the system. The joint distribution factorizes as follows:

$$p_{\boldsymbol{\theta},\psi}(\mathcal{D}, \mathbf{Z}) = \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} | \mathbf{z}_{\mathrm{t}^{(i)}}) \prod_{t=0}^{T-1} p_{\psi}(\mathbf{z}_{t+1} | \mathbf{z}_t), \tag{4.1}$$

where $\mathrm{t}^{(i)}$ is the publication timestep of document $\boldsymbol{x}^{(i)}$ and $\psi = (\boldsymbol{w}, \sigma^2, \mathbf{z}_0)$ are the temporal parameters, and $\mathbf{Z} \in \mathbb{R}^{T \times d_z}$ is the matrix containing latent vectors $\mathbf{z}_t$. $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is parameterized by an LSTM where the latent state $\mathbf{z}$ is concatenated to every word embedding vectors.

4.4.2 Inference

Learning the generative model in Equation 4.1 requires to infer the latent variables $\mathbf{z}^t$. In Bayesian inference, it is done by estimating their posterior $p_{\boldsymbol{\theta},\psi}(\mathbf{Z}|\mathcal{D}) = \frac{p_{\boldsymbol{\theta},\psi}(\mathcal{D},\mathbf{Z})}{\int p_{\boldsymbol{\theta},\psi}(\mathcal{D},\mathbf{Z})\,\mathrm{d}\mathbf{Z}}$. Unfortunately, the marginalization on $\mathbf{Z}$ requires to compute an intractable normalizing integral. We, therefore, use Variational Inference (VI) and consider a variational distribution $q_{\phi}(\mathbf{Z})$ that factorizes across all timesteps:

$$q_{\phi}(\mathbf{Z}) = \prod_{t=1}^{T} q_{\phi}^{t}(\mathbf{z}_t), \tag{4.2}$$

where q_{ϕ}^{t} are independent Gaussian distributions $\mathcal{N}(\mu_t, \sigma_t^2)$ with diagonal covariance matrices σ_t^2, and ϕ is the total set of variational parameters.

This factorization is possible because recurrent language modeling is an autoregressive task (c.f. Section 4.3) that does not require an auto-encoding scheme.

We are thus able to learn a model with fewer parameters while avoiding common pitfalls associated with variational text auto-encoders, e.g. Kullback-Leibler Divergence (KLD) vanishing (Bowman et al. 2016).

A particularity of our approach is that we consider that several documents can be published at the same timestep. So, to obtain an Evidence Lower Bound (ELBO) $\mathcal{L}(\boldsymbol{\theta}, \psi, \boldsymbol{\phi})$, we adapt the derivation in Krishnan et al. 2017 as follows:

$$
\begin{aligned}
\log p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{D}) &= \log \int_{\mathbf{Z}} p_{\psi}(\mathbf{Z}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathbf{z}_t)\, \mathrm{d}\mathbf{Z} \\
&= \log \int_{\mathbf{Z}} q_{\boldsymbol{\phi}}(\mathbf{Z}) p_{\psi}(\mathbf{Z}) \frac{\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathbf{z}_t)}{q_{\boldsymbol{\phi}}(\mathbf{Z})}\, \mathrm{d}\mathbf{Z} \\
&\geq \int_{\mathbf{Z}} q_{\boldsymbol{\phi}}(\mathbf{Z}) \log \left(p_{\psi}(\mathbf{Z}) \frac{\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathbf{z}_t)}{q_{\boldsymbol{\phi}}(\mathbf{Z})} \right) \mathrm{d}\mathbf{Z} \\
&= \sum_{t=1}^{T} \int_{\mathbf{z}_t} q^t_{\boldsymbol{\phi}}(\mathbf{z}_t) \log p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathbf{z}_t)\, \mathrm{d}\mathbf{z}_t + \int_{\mathbf{z}_{t-1}} q^{t-1}_{\boldsymbol{\phi}}(\mathbf{z}_{t-1}) \log \frac{p_{\psi}(\mathbf{z}_t|\mathbf{z}_{t-1})}{q^t_{\boldsymbol{\phi}}(\mathbf{z}_t)}\, \mathrm{d}\mathbf{z}_{t-1}\, \mathrm{d}\mathbf{z}_t \\
&= \sum_{t=1}^{T} \mathbb{E}_{q^t_{\boldsymbol{\phi}}(\mathbf{z}_t)} \left[\log p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathbf{z}_t) \right] - \mathbb{E}_{q^{t-1}_{\boldsymbol{\phi}}(\mathbf{z}_{t-1})} \left[D_{\mathrm{KL}}(q^t_{\boldsymbol{\phi}}(\mathbf{z}_t) \| p_{\psi}(\mathbf{z}_t|\mathbf{z}_{t-1})) \right], \qquad (4.3)
\end{aligned}
$$

where $\mathcal{D}_t$ is the set of all documents published at timestep t, and the inequality is obtained thanks to the Jensen theorem on concave functions. This ELBO can be classically optimized via stochastic gradient ascent using the re-parametrization trick (Kingma et al. 2014; Rezende et al. 2014).

The posterior factorization presented in Equation 4.2 yields an ELBO that is also factorized in time. We can see that in the KLD only two timesteps, t and $t-1$, are considered, meaning that the transition function $p_{\psi}(\mathbf{z}_t|\mathbf{z}_{t-1})$ is learned by matching pairs of latent states distributions. This factorized ELBO simplifies the training, has every term in Equation 4.3 can be computed in parallel.

Global temporal states coupled with variational distributions independent in time offer several learning and computational advantages compared to the deterministic dynamics learned by the STNN in Chapter 3. The two objective functions have the same structure: a reconstruction term and a dynamic term. It is the dynamic term that differentiates the two models. In the STNN objective function (Equation 3.1), the dynamics term is a sum of quadratic errors between inferred and predicted latent states. Errors on each timestep have the same cost, and if an anomaly or a strong disruption appends in the data at a single timestep, the learning algorithm is likely to modify the transition function, leading to a potential impact on consecutive states.

In the proposed objective function (Equation 4.3), the dynamics is stochastic and is learned with a KLD. This divergence function takes into account the variance of the prior and posterior distributions. We chose to fix the prior variance to a scalar σ^2, which is a hyper-parameter of the model. With this setup, the learning

algorithm can choose to ignore difficult transitions, at a cost depending on σ^2. Instead of changing the dynamics if a disruption occurs in the data, the learning algorithm can simply increase the variance of the posterior at this timestep. This allows the learning algorithm to adapt the stochastic dynamics according to the regularity level of the data. This behavior can be controlled by tuning σ^2. A large σ^2 will allow for great discrepancies between the dynamics and the inferred variables, as the weight of the KLD in the ELBO will decrease. On the other hand, small values of σ^2 will make the objective function closer to Equation 3.1, as the prior will become close to a Dirac function.

4.4.3 Experimental Settings

Models and Baselines

In our experiments, we compare the following models [1]:

- A standard regularized LSTM. This baseline has no temporal component but is currently the state-of-the-art in language modeling.
- The DiffTime model (DT) presented in Rosenfeld et al. 2018 is a deep model that produces temporal word embeddings. They proposed to learn a single set of word embedding which are modified according to a given timestep. To adapt a word embedding to a particular timestep t, they learn non-linear transformations that project the word embedding and the scalar timestep into vectorial spaces of identical dimensions. The two resulting vectors are then multiplied and projected into the word embedding space through a linear mapping.
- The Dynamic Word Embeddings (DWE) model (Bamler et al. 2017) learns Gaussian word embeddings with a probabilistic version of the skip-gram algorithm. In this model, the latent variables are the word and context embeddings matrices $\mathbf{U}_t$ and $\mathbf{V}_t$ which follow a generative model of the form:

$$p(\mathbf{Z}, \mathbf{U}, \mathbf{V}) = \prod_{t=0}^{T-1} p(\mathbf{U}_{t+1}|\mathbf{U}_t) p(\mathbf{V}_{t+1}|\mathbf{V}_t) \prod_{t=1}^{T} \prod_{i,j=1}^{L} p(\mathbf{z}_{ij,t}|\mathbf{u}_{i,t}, \mathbf{v}_{j,t}),$$

where $z_{ij,t}$ is the number of times the word $\boldsymbol{w}_j$ appears in the context of the word $\boldsymbol{w}_i$, and L is the size of the vocabulary. In their work, the priors $p(\mathbf{U}_{t+1}|\mathbf{U}_t)$ and $p(\mathbf{V}_{t+1}|\mathbf{V}_t)$ can be viewed as diffusion processes, implemented as follows:

$$p(\mathbf{U}_{t+1}|\mathbf{U}_t) = \mathcal{N}(\mathbf{U}_t, \sigma_t^2)\mathcal{N}(0, \sigma_0^2),$$

1. Code of the models available at https://github.com/edouardelasalles/drlm

where σ_t^2 and σ_0^2 are hyper-parameter governing the diffusion possess. From this equation, we can see that the prior forces the word representation to stay in the center of the space (second Gaussian), and also stay close to the representation of the previous timestep. Here, deep VI is not used to learn a generative model, but as a principled Bayesian framework to learn and align word embeddings in time.

- The Dynamic Recurrent Language Model (DRLM) proposed in this paper with learned transition function.
- The DRLM-Id model proposed in this paper, where the transition function is replaced by the identity matrix so that $\mathbf{z}^{t+1} \sim \mathcal{N}(\mathbf{z}^t, \sigma^2)$.

For comparison purposes, we adapted the temporal word embedding models DT and DWE for language modeling, by replacing the skip-gram component with an LSTM. More details can be found in Appendix A.

Temporal Settings

The two temporal tasks we are interested in in this thesis are prediction and imputation.

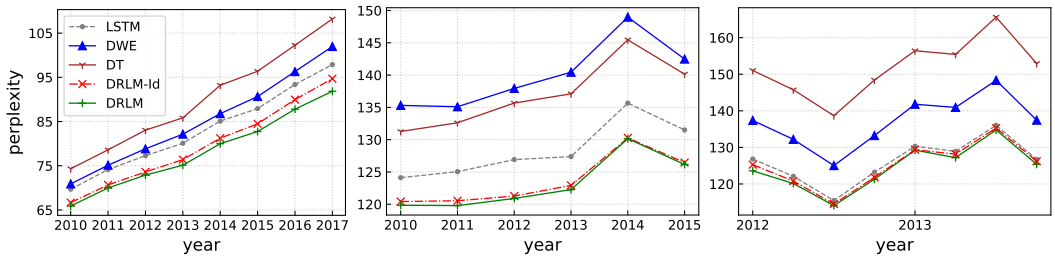
For prediction, we take the first T_p timesteps to train the model. Timesteps $T_p + 1$ to T, with T the total number of timesteps, are used for evaluation. For DRLM, we use the transition model g to predict future states $\mathbf{z}_t$ in time. For DT and DWE we use the embeddings from the last training timestep T_p. Timestep $T_p + 1$ is used for hyper-parameters tuning.

In this part of the manuscript, we do not tackle the typical imputation task, but a variant permitted by the data at hand, that we called modeling. It consists of randomly splitting the corpora into a training (60%), validation (10%), and test (30%) sets for each timestep. It is a task simpler than imputation since we have access to data at each timestep. This task intends to assess the benefit of incorporating temporal information in traditional modeling tasks.

We evaluate the models on language modeling and downstream classification tasks. For language modeling, the evaluation metric is the token level perplexity on the respective test sets. We report the micro-perplexity and the macro temporal perplexity. The micro-perplexity is the global token-level perplexity computed indifferently across timesteps. It is the classical language modeling metric that we use to primarily compare model performances, and in our case writes as follows:

$$\text{Micro-Perplexity}(\mathcal{D}) = \exp\left(-\frac{1}{\sum_{\mathbf{x}\in\mathcal{D}} |\mathbf{x}|} \sum_{\mathbf{x}\in\mathcal{D}} \log p_{\boldsymbol{\theta}}(\mathbf{x})\right),$$

where $p_{\boldsymbol{\theta}}$ is the language model evaluated. For our model, we sample a $\boldsymbol{z}_t \sim q_{\boldsymbol{\phi}}(\mathbf{z}_t)$ for each document and use $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_t)$ for evaluating the perplexity.



(a) S2 (b) NYT (c) Reddit

Figure 4.2 – Perplexity through time for prediction setting.

We also provide the macro perplexity, which is the token-level perplexity computed on each timestep separately and then averaged:

$$\text{Macro-Perplexity}(\mathcal{D}) = \frac{1}{T}\sum_{t=1}^{T} \exp\left(\frac{1}{\sum_{\mathbf{x}\in\mathcal{D}_t}|\mathbf{x}|}\sum_{\mathbf{x}\in\mathcal{D}_t}\log p_{\boldsymbol{\theta}}(\mathbf{x})\right).$$

Since this metric puts the same weight on each timestep, it is possible to see if a model performs consistently across timesteps, even when documents are not evenly distributed in time.

For classification, we report F1 scores for multi-label classification:

$$\text{F1} = \frac{2 \cdot \text{true positive}}{2 \cdot \text{true positive} + \text{false negative} + \text{false positive}},$$

and top1 scores for multi-class classification, which is the proportion of examples on which the model ranks the valid target at the first position.

4.4.4 Results

Prediction

Figure 4.2 shows the evolution of perplexity for prediction. On the three corpora, both DRLM-Id and DRLM beat all baselines. The standard LSTM always performs better than the DWE and DT baselines that systematically overfit. This shows that LSTMs language models are powerful, even without temporal components, and conditioning them is not trivial. Results on Reddit (Figure 4.2c) tend to confirm this observation: performances of LSTM, DRLM-Id, and DRLM are quasi-equivalent, with a gain of 2 points of perplexity for DRLM compared to LSTM. It is a corpus twice larger than the others, with longer sequences. Our analysis is that with sufficient data, and due to the autoregressive nature of textual data, LSTM manages to capture temporal biases implicitly, even without explicit temporal prior.

Algorithm 4.1 Recursive inference

Inputs: Documents $\mathcal{D}_t \ \forall t \in \{T_p + 1, ..., T\}$, parameters $\boldsymbol{\theta}$, $\boldsymbol{\psi}$, and $\boldsymbol{\phi}$ learned on $\mathcal{D}_t \ \forall t \in \{1, ..., T_p\}$
Freezed parameters: $\boldsymbol{\theta}$, $\boldsymbol{\psi}$, variational parameters of $q^t_{\boldsymbol{\phi}}(\mathbf{z}_t)$ for $t \in \{1, ..., T_p\}$
Parameters: variational parameters of $q^t_{\boldsymbol{\phi}}(\mathbf{z}_t)$ for $t \in \{T_p + 1, ..., T\}$
for $t = T_p + 1 \rightarrow T$ **do**
Optimize the variational parameters of $q^t_{\boldsymbol{\phi}}(\mathbf{z}_t)$ by gradient descent on

$$\mathbb{E}_{q^t_{\boldsymbol{\phi}}(\mathbf{z}_t)}\left[\log p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathbf{z}_t)\right] - \mathbb{E}_{q^{t-1}_{\boldsymbol{\phi}}(\mathbf{z}_{t-1})}\left[D_{\mathrm{KL}}(q^t_{\boldsymbol{\phi}}(\mathbf{z}_t)\|p_{\boldsymbol{\psi}}(\mathbf{z}_t|\mathbf{z}_{t-1}))\right]$$

Freeze the variational parameters of $q^t_{\boldsymbol{\phi}}(\mathbf{z}_t)$.
end for

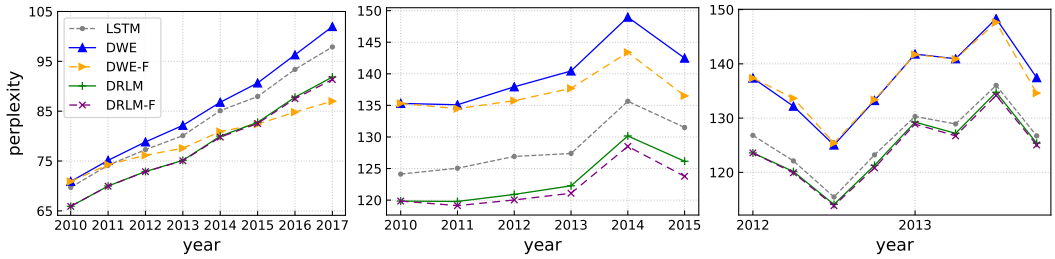
In the S2 corpus, we can see in Figure 4.2a that, while the perplexity of DRLM-Id tends to converge to LSTM's perplexity, DRLM presents consistent improvement through time. On the NYT corpus, while DRLM-Id and DRLM have significant performance gain compared to LSTM (more than 5 points), the difference between the two models is small and vanishes with time. This is explained by the fact that news headlines from NYT are mostly induced by external factors, while scientific publications from S2 are influenced by one another through time.

Recursive Inference

We made the hypothesis that news headlines from NYT are generated mostly by external factors, while scientific publications from S2 are influenced by one another through time, which would explain the absence of performance gain of DRLM compared to DRLM-Id on the NYT.

To validate this hypothesis, we recursively infer the latent states of DRLM. We optimize the variational parameters of every $\mathbf{z}_t$ for $t > T_p$ by maximizing Equation 4.3 according to data from $\mathcal{D}^t$ and states inferred from previous steps. All other parameters remain unchanged. Specifically, we infer $\mathbf{z}_t$ according to $\mathcal{D}^t$ and $\mathbf{z}_{t-1}$. We then evaluate the resulting model at $t+1$, and next, we infer $\mathbf{z}^{t+1}$ according to $\mathcal{D}^{t+1}$ and $\mathbf{z}_t$, evaluate at $t+2$, and so on. Algorithm 4.1 details the procedure. The same process is performed for the variational parameters of DWE. This temporal task is similar in spirit to the filtering task mentioned in the introduction of this thesis. The two resulting models are respectively referred to as DRLM-F and DWE-F in Figure 4.3.

We first observe that the DWE baseline benefits a lot more from recursive inference than DRLM. This is expected since it can adapt each word embedding at each timestep, whereas DRLM-F only infers the distribution of a single vector per timestep. This thus makes DWE-F a good baseline for assessing temporal drift. DRLM-F improves performances on the last timesteps of NYT. It means



(a) S2 (b) NYT (c) Reddit

Figure 4.3 – Perplexity through time with recursive inference. DRLM-F and DWE-F are trained on T_p timesteps, and then their variational parameters are recursively inferred on data at timestep $T_p + \tau$ and evaluated at $T_p + \tau + 1$. The LSTM baseline is displayed for comparison purposes.

that the recursive inference procedure is able to infer latent states that performed better than the predicted ones. It also means that the LM is able to interpret these new variables while being trained on different ones. This is not trivial, given the difficulties of learning conditional language models Variational Auto-Encoders (VAEs) with LSTMs (Bowman et al. 2016; Semeniuta et al. 2017; Yang et al. 2017).

On S2, recursive inference improves performances of DWE, which means that the corpus presents a temporal drift. However, recursively inferred states yield the same performances as predicted ones for DRLM, while on NYT we witness a performance improvement. It can mean that DRLM predicted accurate latent states since recursive inference does not improve results while using future data not seen when performing prediction.

On Reddit, recursive inference does not improve the performances of any of the models and baselines. We interpret that as a lack of temporal drift in the corpus, since no additional data from the future is able to improve prediction performances. Hence, we will not use this corpus in the rest of the manuscript.

To confirm these hypotheses, we plot in Figure 4.4 the latent trajectories of the two components of $\mathbf{z}$ that vary the most through time for DRLM (first row) and DRLM-Id (second row). For DRLM, the inferred points correspond to the means of $q(\mathbf{z}_t)$, and the prior points correspond to the means of $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ for training timesteps. The predicted points for test timesteps are obtained by recursively applying the transition function g from the last training $\mathbf{z}_t$, and the filtered points are those obtained by recursive inference. For DRLM-Id, we only report the inferred points, as there is no transition function to apply (and the prior at each timestep is the state inferred at the previous one).

By comparing the first and second row of Figure 4.4, we first observe that learning a transition function allows the model to learn smoother latent states in time compared to DRLM-Id. This confirms the relevance of our end-to-end learning process, compared to an approach that would learn a transition function

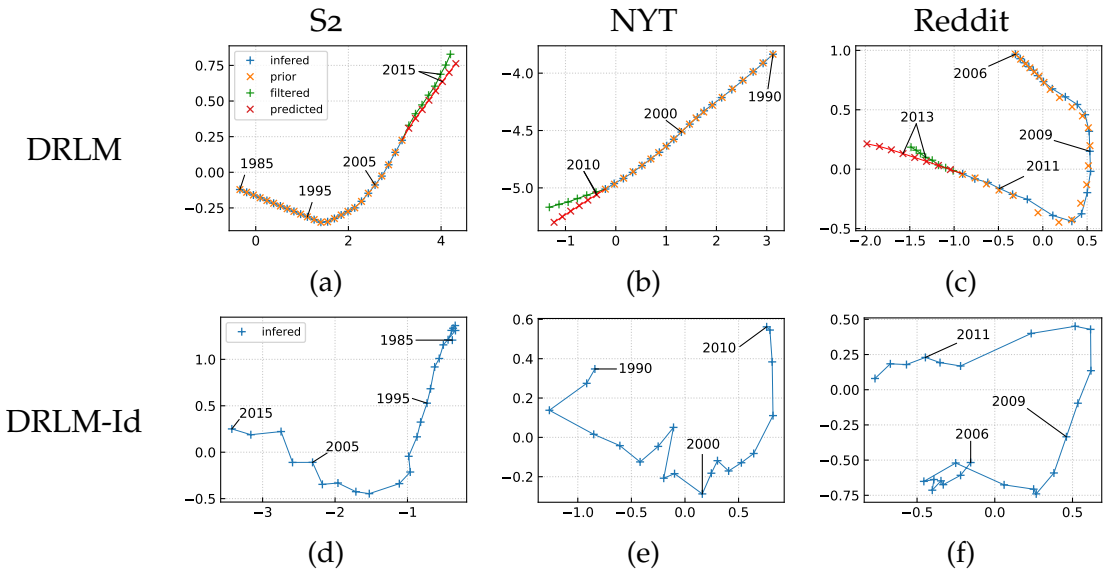


Figure 4.4 – Latent trajectories of the two most varying components of $\mathbf{z}_t$ for the prediction task on the three datasets, for DRLM and DRLM-Id. Each column corresponds to a different corpus. On the first line, latent states are obtained with DRLM, and with DRLM-Id on the second line.

from trajectories inferred by DRLM-Id a posteriori. DRLM automatically organizes states in a smooth fashion, from which extrapolation is easier. On Figure 4.4a, we see that the predicted latent states are very close to the filtered ones, confirming the ability of the transition model to capture and predict global tendencies in the data. On the NYT corpus (Figure 4.4b), we observe that the predicted latent states diverge slightly from the filtered states, which is coherent with the gain in perplexity observed in Figure 4.3b by DRLM-F. On the Reddit corpus, we see that the filtered states are close in time, indicating a slow temporal drift. This is also coherent with the perplexities observed in Figure 4.3c.

Modeling

Table 4.1 presents results for the modeling setup. As for prediction, temporal word embeddings baselines also fail to beat the LSTM baseline. All perplexities are lower since the task is easier, but our models DRLM and DRLM-Id keep their perplexity gain over LSTM.

Text Classification Results

To further evaluate the representations learned by DRLM, we extract its word embeddings augmented with temporal states and use them for text classification. For the DT and DWE baselines, we learned temporal embeddings exactly as

Table 4.1 – Modeling perplexity, where training and testing timesteps are the same.

Model	S2		NYT		Reddit	
	micro	*macro*	*micro*	*macro*	*micro*	*macro*
LSTM	62.8	66.2	109.9	110.4	116.7	123.0
DT	70.7	73.9	125.6	120.4	136.8	147.7
DWE	65.9	69.8	119.9	120.4	129.4	139.6
DRLM-Id	60.6	**61.3**	104.0	104.4	115.5	121.5
DRLM	**60.2**	**61.2**	**103.5**	**103.9**	**114.7**	**120.4**

Table 4.2 – Classification results, with temporal word embeddings for the *prediction* configuration.

Task	Prediction			Modeling		
Corpus	S2 (F1)	NYT (top1)	Reddit (top1)	S2 (F1)	NYT (top1)	Reddit (top1)
LSTM	0.19	35.1	32.0	0.22	41.4	44.0
DT	0.15	19.1	12.5	0.11	17.3	40.9
DWE	0.18	33.4	34.3	0.17	24.8	44.5
DRLM	**0.21**	**41.2**	**38.0**	**0.23**	**44.8**	**45.2**

described in their respective papers, contrary to previous tasks where we used our LSTM LMs adaption of their model. For every classification task, we learn a linear classifier that takes as inputs the average of the embeddings of each sequence, following Grave et al. 2017b and Shen et al. 2018.

Labels are articles' keywords for S2 (multi-label with 400 classes), articles' sections for NYT (mono-label with 28 labels) and subreddits in which posts were submitted for Reddit (mono-label with 60 labels). Classification results for prediction and modeling settings are presented in Table 4.2. DRLM outperforms all baselines. This shows that the representations it learns contain useful information that can be used for downstream tasks such as classification.

Text Generation Through Time

We present here text samples generated by beam search with DRLM trained with the modeling setting. We use starting word triplets that most often appear in the S2 test set as a seed, and we change the latent state through time. Table 4.3 presents generated samples where the latent state evolves from 1985 to 2017. We can see a smooth evolution in vocabulary. Around the 90s, we can see that the language model evolves slowly, as the same sequences are generated 5 years apart

Table 4.3 – Text sequences generated with DRLM conditioned on different timesteps on the S2 corpus. The first three words are uses as seeds, and the samples are generated by beam search with a beam size of 5.

a framework for...	
1985	...shape recovery from images
1995	...shape recovery from images
2005	...automatic evaluation of statistical machine translation
2015	...unsupervised feature selection
2016	...unsupervised learning of deep neural networks
2017	...training deep convolutional neural networks

unsupervised learning of...	
1985	...hidden markov models
1995	...gaussian graphical models
2005	...named entity recognizers
2015	...deep convolutional neural networks
2016	...convolutional neural networks
2017	...generative adversarial networks

a comparison of...	
1985	...smoothing techniques for statistical machine translation
1995	...smoothing techniques for word sense disambiguation
2005	...smoothing techniques for statistical machine translation
2015	...convolutional neural networks for action recognition
2016	...convolutional neural networks for action recognition
2017	...convolutional neural networks for action recognition

in the first set of samples. And we see that the language model starts to evolve quickly from 2015, where references to deep learning begin to appear. In the second set, we even see a reference to Generative Adversarial Model (GAN) on the 2017 sample.

Discussion

In the first part of this chapter, we proposed a dynamic recurrent LM for handling temporal drifts in language. Language evolution dynamics are captured via a learned transition function producing trajectories of temporal states through time. We also learned a transition function, which structures temporal states in time. Experiments on three corpora with various sizes, time scales, and language

levels, showed that our approach beats temporal embeddings baselines in various settings and on downstream classification tasks.

In the proposed model, we learned global latent states by deep VI. We followed a design similar to SSM, which led us to learn our dynamic function by minimizing a KLD on state pairs. We saw in Section 4.4.2 that inferring and predicting distributions rather than points has some learning advantages. Mainly, it allows the learning algorithm to handle disruptions in the data without upsetting the dynamics. However, the STNN and DRLM objectives have a common flow: dynamics is learned by a loss function between latent state pairs only. It is then hard for the model to learn a dynamics that is coherent on the entire length of the temporal phenomenon, as gradients are backpropagated only between pairs of consecutive timesteps.

In the next part, our objective is to incorporate authors in the framework to learn dynamic author representations. This time, we chose to learn a deterministic dynamic model, that allows gradient backpropagation throughout the whole time-period. In this setting, since the latent states are point estimates, the LSTM LM becomes the only source of stochasticity in the model.

4.5 Dynamic Author Representations

In this part, we learn latent representations of authors that evolve through time. We learn latent vectors that represent features specific to textual expression modes of the authors. In order to handle temporal drifts, we propose a dynamic model that updates authors' representations through time in the latent space.

4.5.1 Model

This method is also base on an LSTM network that we condition to an author a and a timestep t through a latent vector $h_{a,t}$. We consider that all the information specific to the author a at time t is contained in this vector. The probability of a document $\boldsymbol{x}$ written by a at time t for an LSTM with parameters θ is defined as follows:

$$P(\boldsymbol{x}|a,t) = P_\theta(\boldsymbol{x}|h_{a,t}) = \prod_{k=0}^{|\boldsymbol{x}|} P_\theta(x_{k+1}|\boldsymbol{x}_{0:k}, h_{a,t}).$$

An overview of our approach is pictured in Figure 4.5.

Depending on the way the condition $h_{a,t}$ is defined for a timestep t and an author a, the model can greatly differ in the dynamics and dependencies it captures.

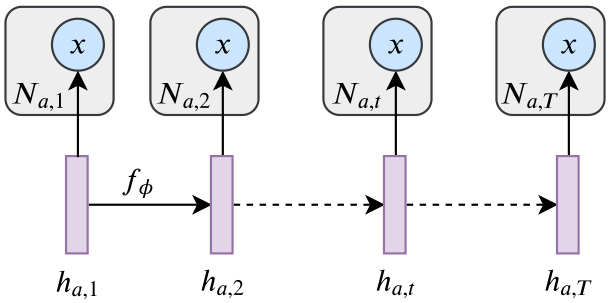


Figure 4.5 – A high-level view of our proposed dynamic language model for an author a. $h_{a,t}$ are the conditioning vectors that evolve through time with a dynamic function f_ϕ. $\boldsymbol{x}$ are text publications at different timesteps and $N_{a,t}$ is the number of texts published by author a at timestep t. The panels surrounding each variable $\boldsymbol{x}$ highlight the fact that several documents ($N_{a,t}$) are modeled conditionally on the same vector $h_{a,t}$.

The general idea of the model is to produce a latent trajectory for each author. A latent trajectory is a sequence of representation vectors $h_{a,t}$ that evolve in time with a function f_ϕ parameterized by ϕ. The general formulation is as follows:

$$h_{a,t} = f_\phi(h_{a,0}, ..., h_{a,t-1}).$$

The formulation is fairly general, and several architectures can fit f_ϕ.

The challenge of learning the $h_{a,t}$ vectors is twofold. First, they should capture features specific to author a that do not change in time. For instance, in the case of a scientific community, the scientific scope of an author (computer science, physics, biology, etc...) usually does not change through the years. And second, it should capture the variations in authors' expression mode and topic evolution through time. The writing style of an author may indeed change through time, and its topics of interest may also change more or less drastically.

To facilitate the learning of static features, a latent vector h_a is learned for each author. These vectors are constant through time and used in various ways in our model. It allows the dynamic function to focus only on variations across timesteps, as described below.

We use a residual architecture for our dynamic function. We chose a Markovian transition function, which only considers the previous representation $h_{a,t-1}$, for the induction of $h_{a,t}$. It appears as a good trade-off between robustness and flexibility. More powerful sequential models, such as RNNs that maintain a memory of the past states, would be prone to overfitting. Indeed, the number of authors and timesteps is usually small and lots of author-timestep pairs are missing. Having a residual function in our dynamics allows us to learn smooth trajectories, as the

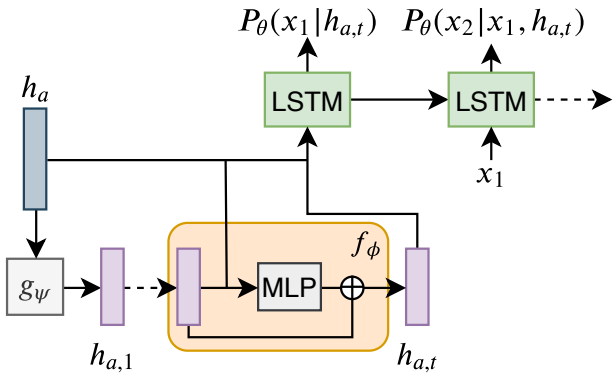


Figure 4.6 – Detailed view of the proposed architecture. The initialization function g_ψ uses the static representation of author a to produce the first latent vector $h_{a,1}$. The residual function f_ϕ is then recursively applied in order to produce $h_{a,t}$, which is used by the LSTM decoder to model a text sequence $\boldsymbol{x}$ written by a at t.

magnitude and direction of the residue can be constrained easily by regularizing ϕ with an ℓ_2 norm. This dynamic function writes as follows:

$$h_{a,t} = h_{a,t-1} + f_\phi(h_{a,t-1}, h_a).$$

In this case, f_ϕ is an Multi-Layer Perceptron (MLP) with Rectified Linear Unit (ReLU) activations. In addition to the previous state, the static representation h_a is also given as input to the MLP in order to encourage different dynamics among authors. Without it, two representations at the same position in the latent space would have the same next state, and hence the same following dynamics. h_a is also used to compute the initial vector $h_{a,1}$ through a specific MLP, g_ψ.

Finally, $h_{a,t}$ vectors are concatenated to the static author representations h_a to form the conditioning vectors that are fed to the LSTM decoder. The decoder is able to capture general language structure, like syntax and grammar, and use the conditioning vectors to adapt its internal dynamic to a specific author at a specific timestep. A detailed view of the described architecture is pictured in Figure 4.5.1.

4.5.2 Experimental Setup

Model and Baselines

We compare the following models:

- **LSTM**: a classical LSTM decoder (no conditioning on the publication time or the authors). We use this model to assess the gain in performances of our model and other baselines.

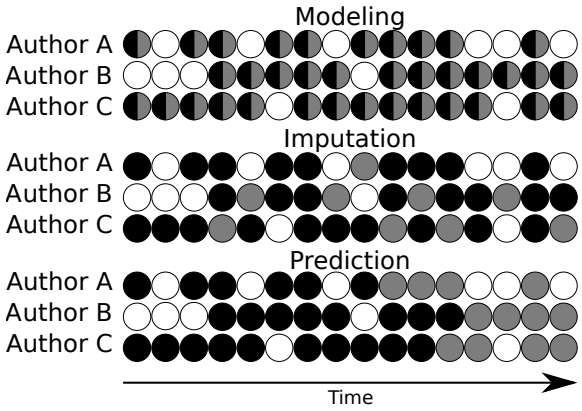


Figure 4.7 – Illustration of our three tasks. A, B, and C are three authors, and each column a timestep. Each circle represents the set of documents (possibly empty) published by a given author at a given timestep. Black circles are training data, grey circles test data, and white circle missing data. Validation data were omitted for simplification purposes.

- **LSTM-A**: an LSTM decoder conditioned on authors embeddings. Only h_a is given as the start token of the LSTM decoder. This baseline allows us to assess the performances of our temporal component.
- **LSTM-iAT**: an LSTM decoder conditioned on authors and time with vectors $h_{a,t}$ that are free parameters to be learned (no dynamics and no constraints on successive vectors). It is the most naive way to condition a language model on authors and time.
- **LSTM-AT**: similar to LSTM-iAT, but where an ℓ_2 regularization between consecutive vectors is applied during learning in order to structure the embedding space. It is a robust baseline, but without a dynamical module to predict representations.
- **Ours** [2]: the model described in Section 4.5.1.

Evaluation and Tasks

Once again, we evaluate the proposed model on prediction and imputation tasks. Since authors are now involved, the definition of the tasks slightly differs from the previous part. A visual representation of the different temporal settings is shown in Figure 4.7.

For prediction, we split the data in time relatively to each author, so that each author a the same ration of publications in the different folds. Since every author

2. code available at https://github.com/edouardelasalles/dar

Table 4.4 – Perplexity on the Semantic Scholar corpus.

	Modeling		Imputation		Prediction	
Models	micro	macro	micro	macro	micro	macro
LSTM	53.8 ± 0.1	65.0 ± 0.4	57.4 ± 0.1	71.5 ± 0.2	80.7 ± 0.2	83.0 ± 0.5
LSTM-A	48.0 ± 0.1	56.8 ± 0.7	52.7 ± 0.1	63.9 ± 0.5	77.2 ± 0.3	$\mathbf{77.8 \pm 0.9}$
LSTM-iAT	54.3 ± 0.1	68.2 ± 0.8	61.3 ± 2.8	77.1 ± 4.7	83.7 ± 0.2	88.0 ± 0.9
LSTM-AT	47.7 ± 0.1	55.4 ± 0.2	52.3 ± 0.1	62.9 ± 0.3	77.2 ± 0.1	$\mathbf{77.3 \pm 1.3}$
Ours	$\mathbf{46.7 \pm 0.1}$	$\mathbf{53.3 \pm 0.2}$	$\mathbf{51.2 \pm 0.1}$	$\mathbf{60.2 \pm 0.2}$	$\mathbf{74.3 \pm 0.2}$	$\mathbf{77.5 \pm 1.2}$

Table 4.5 – Perplexity on the New York Times corpus.

	Modeling		Imputation		Prediction	
Models	micro	macro	micro	macro	micro	macro
LSTM	112.4 ± 0.2	112.9 ± 0.2	108.8 ± 0.1	109.4 ± 0.2	114.5 ± 0.2	110.1 ± 0.2
LSTM-A	100.1 ± 0.2	100.7 ± 0.2	100.7 ± 0.1	101.3 ± 0.2	113.1 ± 0.3	108.3 ± 0.3
LSTM-iAT	108.9 ± 0.3	110.0 ± 0.4	135.8 ± 0.6	136.6 ± 0.6	121.0 ± 0.6	115.9 ± 0.5
LSTM-AT	97.3 ± 0.1	97.9 ± 0.1	98.9 ± 0.2	99.5 ± 0.2	113.1 ± 0.2	108.3 ± 0.2
Ours	$\mathbf{97.1 \pm 0.1}$	$\mathbf{97.7 \pm 0.1}$	$\mathbf{98.2 \pm 0.3}$	$\mathbf{98.7 \pm 0.2}$	$\mathbf{110.8 \pm 0.4}$	$\mathbf{106.5 \pm 0.3}$

has not the same publication rate, the train set stops at different steps for different authors, as depicted in Figure 4.7.

The modeling task, however, is constructed in the same way as in the previous part: a random train/validation/test split between documents.

In this part, we add an imputation task. For this task, we hide all documents published at randomly chosen timesteps for each author in the train set. For each author, different timesteps are kept. This means that all documents written by author a at time t are either in the train, validation, or test set.

LSTM-iAT and LSTM-AT baselines are not equipped to predict latent representations. So, when evaluating documents published by author a at timesteps t where no document was visible during training, the latent representation $h_{a,t'}$ is used, with $t' < t$ the most recent timesteps where documents were present during training. For our method, the dynamic function f_ϕ is used to predict the representation $h_{a,t}$.

4.5.3 Results

Temporal Language Modeling

Table 4.4 shows the results on the S2 corpus, and Table 4.5 on NYT. The Reddit Corpus is not included in this part since we saw in the previous section that it contains nearly no temporal drift, and hence is less suited for this task. On all

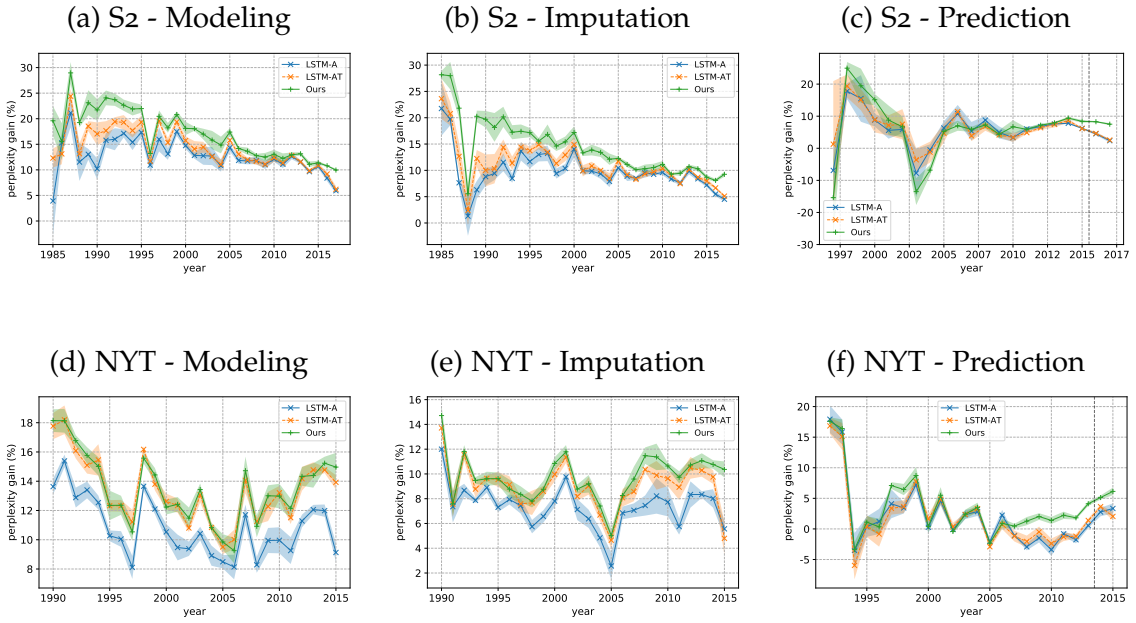


Figure 4.8 – Perplexity gain w.r.t. the LSTM baseline through time for the S2 (top row) and NYT (bottom row) corpora (higher is better). The LSTM-iAT baseline is not displayed because it is often significantly worse than the vanilla LSTM, as shown in Table 4.4 and Table 4.5. The black vertical line on the predictions plots represents the point in time from which no documents were seen in the training sets.

tasks and both corpora, our method is significantly better than all the baselines, in micro and macro perplexity. As expected, taking into account authors in a language model improves its performances. But incorporating time into the model is not trivial. LSTM-iAT has consistently worse performances than the vanilla LSTM, except on NYT modeling. Indeed, this baseline tends to overfit, as it has no temporal regularization. In that case, each vector $h_{a,t}$ allows the model to over-specialize itself on texts from the corresponding author a and time t.

On S2, LSTM-AT, the temporally regularized version of LSTM-iAT, beats LSTM-A by a small margin (0.2 to 0.4 perplexity points), while our model consistently beats it by 1 to 3 perplexity points, indicating that our dynamic function is more efficient at regularizing the latent representation on this corpus. On NYT, our method has performances similar to LSTM-AT on the modeling task and gains 0.7 points on the imputation task. On the more challenging prediction task, on both S2 and NYT, our model beats LSTM-AT with the greatest perplexity gain across all tasks. We also notice that on this task, LSTM-A and LSTM-AT have the same performances on both corpora. This indicates that our dynamic module is able to accurately predict future states, even at unseen timesteps.

Table 4.6 – Ablation study of the dynamic function f_ϕ. Results are in micro-perplexity. Last raw correspond to our full model, as considered in previous experiments.

	S2	NYT
ResNet	47.8 ± 0.2	100.0 ± 0.2
+ AdaDyn	48.0 ± 0.5	97.9 ± 0.3
+ StatCond	46.9 ± 0.1	97.3 ± 0.2
+ AdaDyn + StatCond	$\mathbf{46.7 \pm 0.1}$	$\mathbf{97.1 \pm 0.1}$

To analyze more specifically the results through time, we show in Figure 4.8 the gain in perplexity over the vanilla LSTM through time. For modeling and imputation on S2 (Figure 4.8a and Figure 4.8b), we can see that our method has the highest gain on every timestep. The gain is more important on the first timesteps, which contains far fewer documents than the last ones. It shows that there is a temporal drift in the token distribution and that our model is able to capture it more accurately than a more naive approach. For the same tasks on NYT, we see that LSTM-AT results and ours are similar across timesteps, except for the last ones, where our model maintains the same level of perplexity gain while LSTM-AT tends to fall.

For the prediction task (Figure 4.8c and Figure 4.8f), we observe similar performances for all models on both corpora. It can be explained by the low number of documents for S2, and the difficulty of the task. On the last timesteps, however, our model shows a clear gain over the baselines. On S2, the training set contains no documents published at the 2 last timesteps, which is symbolized by the black vertical line in the figure. The low variance and the significant performance gain of our model on these two timesteps indicate that the dynamic module of our model is able to extrapolate at unseen timesteps. On NYT, our model has better results in the second half of the time period. The poor results of LSTM-AT on this task are due in part to the fact that it does not have a dynamic component, like our model. Instead, the last learned representation vector is used to condition the LSTM on future timesteps. And since the presentations are learned independently for each author at each timestep, strong regularization is required to prevent overfitting when performing prediction.

Ablation Study

In Section 4.5.1, we proposed to use a static representation vector h_a in our model. This vector is used in two ways:

- Adaptive Dynamic (AdaDyn): as an input of the dynamic function to adapt its behavior depending on the current author.

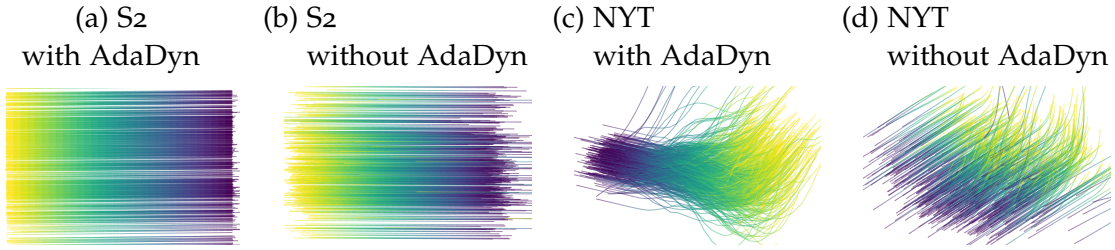


Figure 4.9 – PCA of the latent trajectories $h_{a,t}$ for Semantic Scholar (S2) and New York Times (NYT) with and without AdaDyn. Colors represent time: dark at the first timestep to light as the last.

- Static Conditioning (StatCond): to conditioned the LSTM on the current author independently of time. The objective is to relax $h_{a,t}$ to allow it to focus more on temporal variations.

To assess the contribution of these two features into the final results, we performed an ablation study where each feature combination is removed. The results are shown in Table 4.6. For both corpora, it is the addition of the two features together that yields the best results.

StatCond always increases the performances significantly, as it helps the dynamic module to focus on drifts since authors' information that do not change in time is not required to be carried by the dynamic representation through time.

On S2, contrary to NYT, the AdaDyn alone does not improve performances of the base ResNet. It means that on this corpus the network does not need to learn individual dynamics for each author, but only a global drift.

In the next section, we analyze the learned latent trajectories to confirm this behavior.

Latent Trajectories Visualization

To gain a better understanding of our model behavior, we investigate the temporal author representations learned by our model. All the visualizations in this section were extracted from a model learned on the modeling task.

To visualize the latent trajectories, we performed Principal Component Analysis (PCA) on the representations and pictured them in Figure 4.9. On NYT, we can see that removing the AdaDyn component (Figure 4.9d) yields parallel trajectories, that all of them drift together in time. On the other hand, with AdaDyn (Figure 4.9c), the dynamic function is free to learn a different dynamic for each author, and we see that the representations drift together in time, but also relatively to each other. On S2 on the other hand, with (Figure 4.9a) or without (Figure 4.9b) AdaDyn, the latent trajectories move as one block. It illustrates the results of the

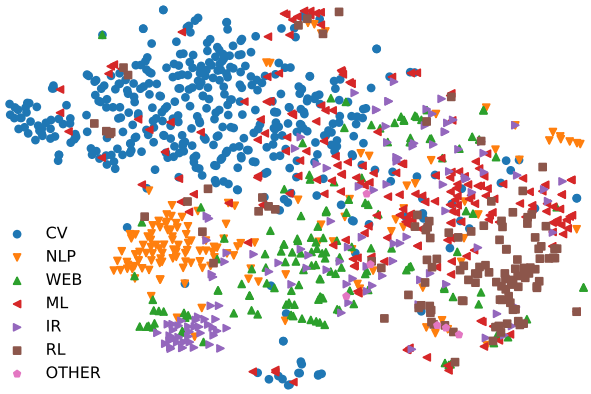


Figure 4.10 – t-SNE visualization of the static representations h_a on the S2 corpus.

ablation study, where we saw that AdaDyn did not improve the results over the ResNet alone on this dataset.

Latent Space Analysis

Here, we provide a more detailed analysis of the latent representations learned on the S2 corpus. Since we just saw that the latent trajectories in S2 do not vary relatively to each other, we focus here on community-level phenomena.

We begin by plotting in Figure 4.10 a t-SNE visualization of the static vectors h_a. The labels in this visualization are obtained thanks to key-words associated to each paper in the S2 dataset, that are interpreted as topics. We manually clustered the labels into 6 general machine learning categories: Computer Vision (CV), Natural Language Processing (NLP), WEB, Machine Learning (ML), Information Retrieval (IR), and Reinforcement Learning (RL). We also put a category OTHER for authors that do not fit in these categories. We label the authors with the most represented category among their publications. We see on the figure that authors from the CV and NLP communities are distinctly clustered. Next to the NLP cluster, we notice a small IR cluster. Next to these two clusters are several authors from the WEB community. RL authors have their own cluster on the right, though less distinct from the others. And finally, the machine learning authors are spread across all the space, which is expected because the category is very broad since the corpus contains only machine learning papers. It indicates that our static vectors capture semantic information about authors.

We further analyze the learned trajectories on S2 by examining cosine similarities between authors in the latent space. We show in Figure 4.11 the average cosine similarity between authors through time. First, we can see that all authors follow the same trend. It was expected since we saw in Figure 4.9a that all authors seem to follow the same dynamics. On the first timesteps, all representations

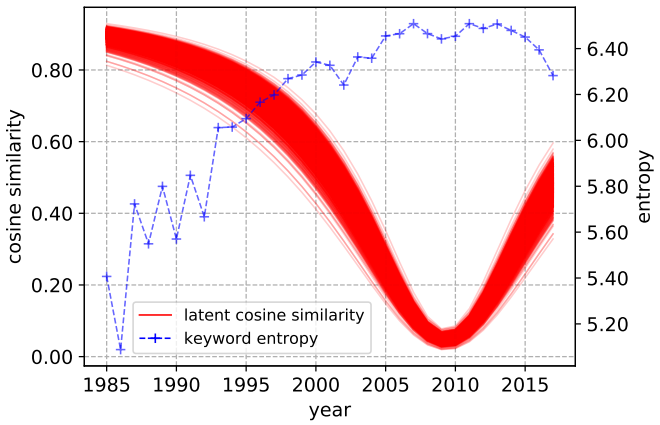


Figure 4.11 – Evolution of latent vectors. Red lines correspond to the averaged cosine similarity between authors in the latent space in the S2 corpus. The blue dotted line is the entropy of keywords at each timestep.

are very similar, with a cosine similarity around 0.9. Since there are only a few documents published at these timesteps, and because of the weight decay on h_a, all representations tend to regroup in the same place, preventing overfitting. The average similarity then drops to 0, as the model learns to drive away each representation to better fit them to each author. And then, after 2009, the average similarities go up to and reach 0.5 on the last timestep. This sudden augmentation in global similarity cannot be explained by the quantity of data, as the last 6 timesteps contain 50% of the documents in the corpus. Another hypothesis is that global diversity among authors diminishes. To illustrate this, we plot the entropy of articles' keywords through time, that we interpret as the diversity of subjects studied in the community. The entropy is plotted in blue in Figure 4.11, and we can see that, symmetrically to the cosine similarities, the entropy of keywords increases from 1985 to 2010 approximately, and then begins to drop. This drop of entropy indicates that the diversity of topics also drops, and is translated by our model in an augmentation of the average similarity between authors.

Data Samples

Here, we present samples generated by our model trained on Semantic Scholar for the modeling task. Each sample is generated by beam search with a beam of size 5 and is seeded with different word triplets that often appear in the corpus.

We conditioned the LSTM decoder of our model to authors randomly sampled, at several timesteps. The samples are presented in Table 4.7. Each table from A to D corresponds to a word triplet seed and each column from 1 to 3 to an author. Note that authors are different between blocks, and the author of A1 is not the author of B1.

Table 4.7 – Samples generated from our model for different authors through time. Text sequences are generated by feeding the first three words displayed in bold at the top of each block. The samples were obtained by beam search with a beam size of 5.

	1	2	3
A		**semi - supervised**...	
1985	...learning	...learning in the presence of noise	...learning of object categories
1990	...learning with the em algorithm	...learning of linear models	...learning of object categories
1995	...learning with a probabilistic model	...learning of probabilistic models	...image segmentation
2000	...learning with kernels	...learning with gaussian processes	...segmentation of 3d objects
2005	...learning with pairwise constraints	...learning for text classification	...segmentation of 3d human motion
2010	...learning with pairwise constraints	...learning for text classification	...multi - view face recognition
2015	...learning with deep neural networks	...multi - task learning	...convolutional neural networks
2016	...learning with deep neural networks	...learning with deep neural networks	...convolutional neural networks
2017	...learning with deep neural networks	...deep learning	...convolutional neural networks
B		**a study of**...	
1985	...image segmentation	...knowledge compilation	...word sense disambiguation
1990	...the fundamental matrix	...knowledge compilation	...word sense disambiguation
1995	...multi - view stereo	...bayesian networks	...statistical machine translation
2000	...image segmentation algorithms	...probabilistic logic programming	...statistical machine translation
2005	...multi - view face recognition	...probabilistic models for relational learning	...statistical machine translation
2010	...energy minimization algorithms	...probabilistic models for relational learning	...statistical machine translation
2015	...modern inference techniques for structured prediction	...variational bayesian inference	...statistical machine translation systems
2016	...modern inference techniques for structured prediction	...variational bayesian inference	...neural machine translation systems
2017	...deep convolutional neural networks	...variational bayesian inference	...neural machine translation systems
C		**real - time**...	
1985	...visual tracking	...visual tracking	...visualization of the web
1990	...visual tracking	...visual tracking	...multi - view stereo
1995	...visual tracking	...visual tracking	...time - series classification
2000	...multi - view clustering	...visual tracking	...time series classification
2005	...collaborative filtering	...facial expression recognition	...time series classification
2010	...bidding in display advertising	...visual tracking using deep learning	...time series classification
2015	...bidding in display advertising	...visual tracking with deep neural networks	...time series forecasting
2016	...bidding in display advertising	...facial expression recognition	...time series forecasting
2017	...bidding in display advertising	...visual tracking with deep neural networks	...time series forecasting
D		**a framework for**...	
1985	...learning to rank	...qualitative simulation	...learning to rank
1990	...learning to rank	...multi - agent reinforcement learning	...learning to rank
1995	...learning to rank	...multi - agent reinforcement learning	...learning to rank
2000	...parsing natural language	...multi - agent reinforcement learning	...learning to rank
2005	...parsing natural language	...multi - agent reinforcement learning	...learning to rank
2010	...multi - task learning	...multi - target tracking	...learning to rank
2015	...multi - task learning	...multi - target tracking	...learning to rank
2016	...multi - task learning	...multi - target tracking	...learning to rank
2017	...recurrent neural networks	...deep reinforcement learning	...learning to rank

We first notice that samples are smooth in time in the text space. We also see different speeds of variation between the different authors. For instance, the samples D3 (bottom right) are always the same at each timestep, while the B1 samples vary rapidly. Generally, we can see that our model tends to generate titles related to deep neural networks at the last timesteps of every author (`recurrent neural networks`, `deep reinforcement learning`, `deep convolutional neural networks`, etc...). It is consistent with the increase in average author similarity found in Figure 4.11. We also see that samples for a particular author across time tend to refer to the same sub-field (e.g. computer vision or natural language processing), which is also consistent with dynamics observed in section 4.11.

Future Works

We present here some possible future works from an NLP point of view. A first direction is toward explicitly discovering relationships between authors, by incorporating our STNN framework. This would allow us to explicitly capture relations between authors, and study their evolution through time. In this work, we did not address the fact that new words appear at future timesteps. Handling out-of-vocabulary words are of major concern in NLP, and predicting new words is even more challenging. Ongoing works focus on learning to predict future sub-words combinations using byte pair encoding (Sennrich et al. 2016). Recently, a new kind of language model architecture based on transformer networks (Devlin et al. 2019; Radford et al. 2019) achieved state-of-the-art results in various NLP tasks. Integrating it and analyzing its effects in our framework is an interesting and promising research direction.

4.6 Conclusion

In this chapter, we explored the incorporation of a temporal component into different language models. We first proposed a dynamic recurrent language model that handles temporal drifts in language. In this model, language evolution dynamic is captured in a latent space with global latent variables that are constrained by a learned transition function. We then propose to also take into account authors, since their individual modes of expression also change through time.

For this last task, we propose a deterministic dynamic model that completely define the latent trajectories of authors representation, compared to our first model where the variationally inferred latent states were free to violate the learned dynamics. However, by using a deterministic dynamics, we loose stochasticity, which hurt performances on certain tasks. In the next chapter, we propose a model that has both features: it is a stochastic model with a deterministic

part. The stochastic aspect allows the model to generate diverse futures, and the deterministic aspect allows gradient backpropagation through entire sequences.

CHAPTER 5

STOCHASTIC PREDICTION OF VIDEOS

Contents

Chapter abstract

Temporal phenomena are often stochastic, which needs to be taken into account to build better prediction models. In this chapter, we propose a stochastic sequential generation model based on Variational Auto-Encoders (VAEs), and applied it to video prediction. Contrary to previous chapters, we consider here a temporal setup where we have several videos of fixed length sampled from the same distribution. This allows us to propose to amortize the inference process with a Neural Network (NN). The proposed model is based on residual updates of a latent state and is motivated by discretization schemes of differential equations. This first-order principle naturally models video dynamics and allows us to build a simpler and more interpretable model compared to the autoregressive state-of-the-art models. We achieve competitive results with these models on 4 datasets with a lighter fully latent model.

- Jean-Yves Franceschi*, Edouard Delasalles*, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (2020). "Stochastic Latent Residual Video Prediction". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*.

5.1 Introduction

In this chapter, we focus on the prediction task by tackling a challenging component of temporal data: their temporal stochasticity. Indeed, in many real-world time series, the same series can have different and diverse futures. This is particularly the case in video data, that is our subject of study in this chapter.

Being able to predict the future of a video from a few conditioning frames in a self-supervised manner has many applications in fields such as reinforcement learning (Gregor et al. 2019) or robotics (Babaeizadeh et al. 2018). More generally, it challenges the ability of a model to capture visual and dynamic representations of the world. Video prediction has received a lot of attention from the computer vision community. However, most proposed methods are deterministic, reducing their ability to capture video dynamics, which are intrinsically stochastic (E. Denton et al. 2018).

Most state-of-the-art approaches are based on image-autoregressive models (E. Denton et al. 2018; Babaeizadeh et al. 2018), built around Recurrent Neural Networks (RNNs). However, as mentioned in the related work of this thesis (c.f. Section 2.4.3), performances of their temporal models innately depend on the capacity of their encoder and decoder, as each generated frame has to be re-encoded in a latent space. Such autoregressive processes induce a high computational cost, and strongly tie the frame synthesis and temporal models, which may hurt the performance of the generation process and limit its applicability (Gregor et al. 2019; Rubanova et al. 2019).

An alternative approach consists in separating the latent dynamics from frame synthesis process, which are independently decoded from the latent space. In addition to removing the aforementioned link between frame synthesis and temporal dynamics, this is computationally appealing when coupled with a low-dimensional latent-space. Moreover, such models can be used to shape a complete representation of the state of a system, e.g. for reinforcement learning applications (Gregor et al. 2019), and more interpretable than autoregressive models (Rubanova et al. 2019). Yet, these State Space Models (SSMs) are more difficult to train as they require non-trivial latent state inference schemes (Krishnan et al. 2017) and a careful design of the dynamic model (Karl et al. 2017). This leads most successful SSMs to only be evaluated on small or artificial toy tasks.

In this chapter, we propose a novel stochastic dynamic model for the task of video prediction, which successfully leverages structural and computational

advantages of SSMs that operate on low-dimensional latent spaces. The dynamic component determines the evolution through residual updates of the latent state, conditioned on learned stochastic variables. This formulation allows us to implement an efficient training strategy and process in an interpretable manner complex high-dimensional data such as videos. This residual principle can be linked to recent advances relating to residual networks and Ordinary Differential Equations (ODEs) (Chen et al. 2018). This interpretation opens new perspectives such as generating videos at different frame rates, as demonstrated in our experiments. Overall, this approach outperforms current state-of-the-art models on the task of stochastic video prediction, as demonstrated by comparisons with competitive baselines on representative benchmarks.

5.2 Video Prediction in Computer Vision

In this chapter, we use the stochastic video prediction task as a proxy to study stochastic generation processes. But since the task is tackled by many previous works, we begin this chapter by referencing the principal body of work related to this task.

Inspired by prior sequence generation models using RNNs, a number of video prediction methods (Srivastava et al. 2015; Villegas et al. 2017; Wichers et al. 2018) rely on Long Short-Term Memories (LSTMs) (Hochreiter et al. 1997), or, like Ranzato et al. 2014 and Jia et al. 2016, on derived networks such as ConvLSTMs (X. Shi et al. 2015) taking advantage of Convolutional Neural Networks (CNNs). Indeed, computer vision approaches are usually tailored to high-dimensional video sequences and propose domain-specific techniques as they often use pixel-level transformations and optical flow (X. Shi et al. 2015; Walker et al. 2015; Finn et al. 2016; Jia et al. 2016; Vondrick et al. 2017; Liang et al. 2017; Liu et al. 2017; Lotter et al. 2017; C. Lu et al. 2017; Fan et al. 2019) that help to produce high-quality predictions. Such predictions are, however, deterministic, thus hurting their performance as they fail to generate sharp long-term video frames (Babaeizadeh et al. 2018; E. Denton et al. 2018). Following Mathieu et al. 2016, some works proposed to use an adversarial loss (Goodfellow et al. 2014) on their model's predictions to sharpen the generated frames (Vondrick et al. 2017; Liang et al. 2017; C. Lu et al. 2017; Xu et al. 2018). Nonetheless, adversarial losses are notoriously hard to train, and lead to mode collapse, preventing diversity of generations.

Some approaches rely on exact likelihood maximization, using pixel-level autoregressive generation (Oord et al. 2016b; Kalchbrenner et al. 2017) or normalizing flows through invertible transformations between the observation space and a latent space (Kingma et al. 2018; Kumar et al. 2019). However, they require careful design of complex temporal generation schemes manipulating high-dimensional

data, thus inducing a prohibitive temporal generation cost. More efficient continuous models rely on Variational Auto-Encoders (VAEs) for the inference of low-dimensional latent state variables. Except Xue et al. 2016 who learn a one-frame-ahead VAE, they model sequence stochasticity by incorporating a random latent variable per frame into a deterministic RNN-based image-autoregressive model. Babaeizadeh et al. 2018 integrate stochastic variables into the ConvLSTM architecture of Finn et al. 2016. Concurrently with J. He et al. 2018, E. Denton et al. 2018, with Castrejón et al. 2019 in a follow-up, use a prior LSTM conditioned on previously generated frames in order to sample random variables that are fed to a predictor LSTM. Finally, Lee et al. 2018 combine the ConvLSTM architecture and this learned prior, adding an adversarial loss on the predicted videos to sharpen them at the cost of a diversity drop. Concurrently to our work, Minderer et al. 2019 propose to use the autoregressive VRNN model (Chung et al. 2015) on learned image key-points instead of raw frames. While this change could mitigate the aforementioned problems, the extent of such mitigation is unclear.

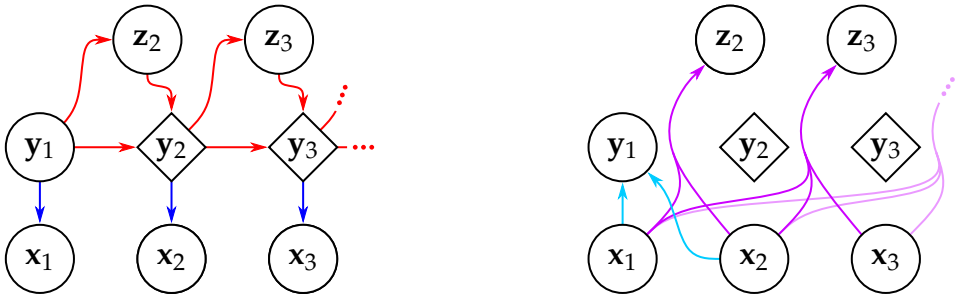
5.3 Model

We consider the task of stochastic video prediction, consisting of approaching, given some conditioning video frames, the distribution of possible future frames given this conditioning. To this end, we proposed a novel temporal latent residual model. This model has two components: one for the dynamics and the other for the content. For the former, one learns latent vectors and the transition function that represent the complete dynamics of a video sequence. The state vectors are stochastically updated by deterministically computing their next value using auxiliary random variables, learned by Variational Inference (VI). The content component of the model is used to model the specificities of video data. For instance, backgrounds are often static, and should not be part of the latent dynamics. The content component is a learned static content variable (E. L. Denton et al. 2017; Li et al. 2018), computed from sampled subsets of frames at training time to avoid any temporal leak in this static variable.

5.3.1 Latent Residual Dynamic Model

Let $\mathbf{x}_{1:T}$ be a sequence of T video frames. We model their evolution by introducing latent variables $\mathbf{y}$ that are driven by a dynamic temporal model. Each frame $\mathbf{x}_t$ is then generated with a decoder from the corresponding latent state $\mathbf{y}_t$ only, making the dynamics independent from the decoding process.

We propose to model the transition function of the latent dynamic of $\mathbf{y}$ with a stochastic residual network. State $\mathbf{y}_{t+1}$ is chosen to be deterministically dependent on the previous state $\mathbf{y}_t$, conditionally to an auxiliary random variable $\mathbf{z}_{t+1}$. These



(a) Generative model p. (b) Inference model q.

Figure 5.1 – Proposed generative and inference models. Diamonds and circles represent, respectively, deterministic and stochastic states.

auxiliary variables encapsulate the randomness of the video dynamics. They have a learned factorized Gaussian prior that depends on the previous state only. The model is depicted in Figure 5.1a, and defined as follows:

$$\begin{cases} \mathbf{y}_1 \sim \mathcal{N}(\mathbf{0}, I), \\ \mathbf{z}_{t+1} \sim \mathcal{N}\big(\mu_{\boldsymbol{\theta}}(\mathbf{y}_t), \sigma_{\boldsymbol{\theta}}(\mathbf{y}_t) I\big), \\ \mathbf{y}_{t+1} = \mathbf{y}_t + f_{\boldsymbol{\theta}}(\mathbf{y}_t, \mathbf{z}_{t+1}), \\ \mathbf{x}_t \sim \mathcal{G}\big(g_{\boldsymbol{\theta}}(\mathbf{y}_t)\big), \end{cases} \tag{5.1}$$

where $\mu_{\boldsymbol{\theta}}$, $\sigma_{\boldsymbol{\theta}}$, $f_{\boldsymbol{\theta}}$, and $g_{\boldsymbol{\theta}}$ are neural networks, and $\mathcal{G}\big(g_{\boldsymbol{\theta}}(\mathbf{y}_t)\big)$ is a probability distribution parameterized by $g_{\boldsymbol{\theta}}(\mathbf{y}_t)$.

In our experiments, $\mathcal{G}$ is a normal distribution with fixed diagonal variance and mean $g_{\boldsymbol{\theta}}(\mathbf{y}_t)$. Note that $\mathbf{y}_1$ is assumed to have a standard Gaussian prior, and, in our VAE setting, will be inferred from conditioning frames for the prediction task, as shown in Section 5.3.3.

The residual update rule takes inspiration in the Euler discretization scheme of differential equations. The state of the system $\mathbf{y}_t$ is updated by its first-order movement, i.e., the residual $f_{\boldsymbol{\theta}}(\mathbf{y}_t, \mathbf{z}_{t+1})$. Compared to a regular RNN, this simple principle makes our temporal model lighter and more interpretable, since trajectories can be computed and followed in a low-dimensional latent space. Equation 5.1, however, differs from a discretized ODE because of the introduction of the stochastic discrete-time variables $\mathbf{z}$. Nonetheless, we propose to allow the Euler step size Δt to be smaller than 1, as a way to make the temporal model closer to a continuous dynamics. The updated dynamics becomes, with $\frac{1}{\Delta t} \in \mathbb{N}$ to synchronize the step size with the video frame rate:

$$\mathbf{y}_{t+\Delta t} = \mathbf{y}_t + \Delta t \cdot f_{\boldsymbol{\theta}}\Big(\mathbf{y}_t, \mathbf{z}_{\lfloor t \rfloor + 1}\Big). \tag{5.2}$$

For this formulation, the auxiliary variable $\mathbf{z}_t$ is kept constant between two integer timesteps. Note that a different Δt can be used during training or testing.

This allows our model to generate videos at an arbitrary frame rate since each intermediate latent state can be decoded in the observation space. This ability enables us to observe the quality of the learned dynamic as well as challenge its ODE inspiration by testing its generalization to the continuous limit in Section 5.4. In the following, we consider Δt as a hyper-parameter. For the sake of clarity, we present the model with $\Delta t = 1$; generalizing to smaller Δt being straightforward as Figure 5.1a remains unchanged.

5.3.2 Content Variable

Some components of video sequences can be static, such as the background or shapes of moving objects. They may not impact the dynamics; we, therefore, model them separately, in the same spirit as E. L. Denton et al. 2017 and Li et al. 2018. We compute a content variable $\mathbf{w}$ that remains constant throughout the whole generation process and is fed together with $\mathbf{y}_t$ into the frame generator. It enables the dynamical part of the model to focus only on movement, hence being lighter and more stable. Moreover, it allows us to leverage architectural advances in neural networks, such as skip connections (Ronneberger et al. 2015), to produce more realistic frames.

This content variable is a deterministic function c_ψ of a fixed number $k < T$ of frames $\mathbf{x}_c^{(k)}$:

$$\mathbf{x}_c^{(k)} = \mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}, \quad \mathbf{w} = c_\psi\left(\mathbf{x}_c^{(k)}\right) = c_\psi\left(\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}\right), \quad \mathbf{x}_t \sim \mathcal{G}\left(g_{\boldsymbol{\theta}}(\mathbf{y}_t, \mathbf{w})\right),$$

where $i_1, \dots, i_k$ represent temporal indices.

This content variable is not endowed with any probabilistic prior, contrary to the dynamic variables $\mathbf{y}$ and $\mathbf{z}$. Hence, the information it contains is not constrained in the loss function (see Section 5.3.3), but only architecturally. To prevent temporal information from leaking in $\mathbf{w}$, we propose to uniformly sample these k frames within $\mathbf{x}_{1:T}$ during training. We also design c_ψ as a permutation-invariant function (Zaheer et al. 2017b), which is done by using an Multi-Layer Perceptron (MLP) fed with the sum of individual frame representations, similarly to Santoro et al. 2017.

This formalism allows us to use skip connections for modeling static background. Skip connections require symmetric encoder and decoder architectures and add as input of each decoder layer the output of its symmetric encoder layer. This design allows direct information flow from the conditioning frames to the generated frames, and help to model static components of videos, like backgrounds.

During testing, $\mathbf{x}_c^{(k)}$ are the last k conditioning frames (usually between 2 and 5). This allows the model to infer content variables with ground truth data closer in time to the predicted one, and hence optimize prediction.

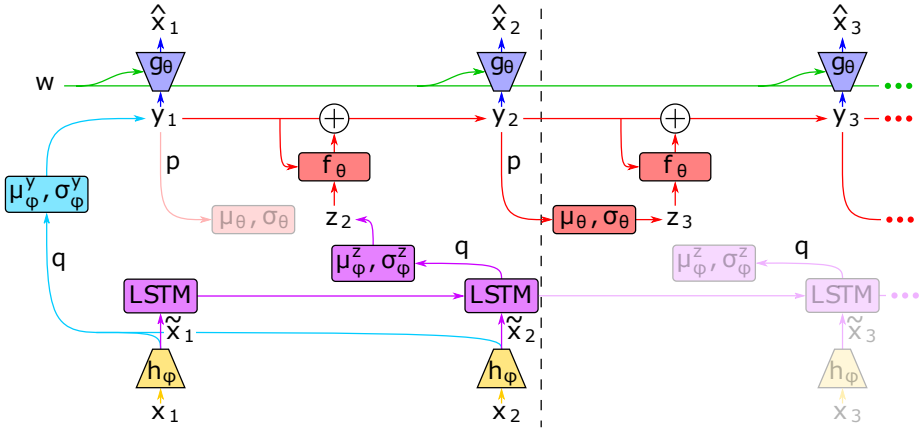


Figure 5.2 – Model and inference architecture on a test sequence. The transparent block on the left depicts the prior, and those on the right correspond to the full inference performed at training time. h_θ and g_θ are deep CNNs, and other named networks are MLPs.

This absence of prior and its architectural constraint allows $\mathbf{w}$ to contain as much non-temporal information as possible while preventing it from containing dynamic information. On the other hand, due to their strong standard Gaussian priors, $\mathbf{y}$ and $\mathbf{z}$ are encouraged to discard any unnecessary information. Therefore, $\mathbf{y}$ and $\mathbf{z}$ should only contain temporal information that could not be captured by $\mathbf{w}$.

Note that this content variable can be removed from our model, yielding a more classical deep state-space model. An experiment in this setting is presented in Appendix D.

5.3.3 Variational Inference and Architecture

Following the generative process depicted in Figure 5.1a, the conditional joint probability of the full model, given a content variable $\mathbf{w}$, can be written as:

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{2:T}, \mathbf{y}_{1:T} \mid \mathbf{w}) = p(\mathbf{y}_1) \prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{y}_t) p(\mathbf{y}_{t+1} \mid \mathbf{y}_t, \mathbf{z}_{t+1}) \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{y}_t, \mathbf{w}), \tag{5.3}$$

where $p(\mathbf{y}_{t+1} \mid \mathbf{y}_t, \mathbf{z}_{t+1}) = \delta\big(\mathbf{y}_t + f_{\boldsymbol{\theta}}(\mathbf{y}_t, \mathbf{z}_{t+1}) - \mathbf{y}_{t+1}\big)$ and δ is the Dirac delta function centered on 0. Thus, in order to optimize the likelihood of the observed videos $p(\mathbf{x}_{1:T} \mid \mathbf{w})$, we need to infer latent variables $\mathbf{y}_1$ and $\mathbf{z}_{2:T}$. This is done by deep VI using the inference model parameterized by ϕ and shown in Figure 5.1b, which

comes down to consider a variational distribution $q_{Z,Y}$ defined and factorized as follows:

$$\begin{aligned} q_{Z,Y} \triangleq & q(\mathbf{z}_{2:T}, \mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{w}) \\ = & q(\mathbf{y}_1 \mid \mathbf{x}_{1:k}) \prod_{t=2}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \delta\big(\mathbf{y}_{t-1} + f_{\boldsymbol{\theta}}(\mathbf{y}_{t-1}, \mathbf{z}_t) - \mathbf{y}_t\big). \end{aligned} \tag{5.4}$$

This yields the following Evidence Lower Bound (ELBO), whose full derivation is given in Appendix B:

$$\begin{aligned} \log p(\mathbf{x}_{1:T} \mid \mathbf{w}) \geq & \mathbb{E}_{(\tilde{\mathbf{z}}_{2:T}, \tilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \tilde{\mathbf{y}}_t, \mathbf{w}) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1 \mid \mathbf{x}_{1:k}) \,\big\|\, p(\mathbf{y}_1)\big) \\ & - \mathbb{E}_{(\tilde{\mathbf{z}}_{2:T}, \tilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=2}^{T} D_{\mathrm{KL}}\big(q(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \,\big\|\, p(\mathbf{z}_t \mid \tilde{\mathbf{y}}_{t-1})\big) \triangleq \mathcal{L}\big(\mathbf{x}_{1:T}; \mathbf{w}, \boldsymbol{\theta}, \phi\big). \end{aligned} \tag{5.5}$$

The sum of Kullback-Leibler Divergence (KLD) expectations implies to consider the full past sequence of inferred states for each timestep, due to the dependence on conditionally deterministic variables $\mathbf{y}_{2:T}$. However, optimizing $\mathcal{L}\big(\mathbf{x}_{1:T}; \mathbf{w}, \boldsymbol{\theta}, \phi\big)$ with respect to model parameters $\boldsymbol{\theta}$ and variational parameters ϕ can be done efficiently by sampling a single full sequence of states from $q_{Z,Y}$ per example, and computing gradients by backpropagation through all inferred variables, using the reparametrization trick. We classically choose $q(\mathbf{y}_1 \mid \mathbf{x}_{1:k})$ and $q(\mathbf{z}_t \mid \mathbf{x}_{1:t})$ to be factorized Gaussian, so that all KLDs can be computed analytically.

We include an ℓ_2 regularization term on residuals $f_{\boldsymbol{\theta}}$ to stabilizes the temporal dynamics of the residual network, as noted by Behrmann et al. 2019 and Rousseau et al. 2019. Given a set of videos $\mathcal{X}$, the full optimization problem, where $\mathcal{L}$ is defined as in Equation 5.5, is given as:

$$\arg\max_{\boldsymbol{\theta}, \phi, \psi} \sum_{\mathbf{x} \in \mathcal{X}} \left[\mathbb{E}_{\mathbf{x}_{\mathrm{c}}^{(k)}} \mathcal{L}\left(\mathbf{x}_{1:T}; c_\psi\left(\mathbf{x}_{\mathrm{c}}^{(k)}\right), \boldsymbol{\theta}, \phi \right) - \lambda \cdot \mathbb{E}_{(\mathbf{z}_{2:T}, \mathbf{y}_{1:T}) \sim q_{Z,Y}} \sum_{t=2}^{T} \left\| f_{\boldsymbol{\theta}}(\mathbf{y}_{t-1}, \mathbf{z}_t) \right\|_2 \right].$$

Figure 5.2 depicts the full architecture of our temporal model, corresponding to how the model is applied during testing. The first latent variables are inferred with the conditioning frames and are then predicted with the dynamics model. In contrast, during training, each frame of the input sequence is considered for inference, which is done as follows.

Firstly, each frame $\mathbf{x}_t$ is independently encoded into a vector-valued representation $\tilde{\mathbf{x}}_t$, with $\tilde{\mathbf{x}}_t = h_\phi(\mathbf{x}_t)$. $\mathbf{y}_1$ is then inferred using an MLP on the first k encoded frames $\tilde{\mathbf{x}}_{1:k}$. Each $\mathbf{z}_t$ is inferred in a feed-forward fashion with an LSTM on the encoded frames. Inferring $\mathbf{z}$ this way experimentally performs better than, e.g., inferring them from the whole sequence $\mathbf{x}_{1:T}$; we hypothesize that this follows from the fact that this filtering scheme is closer to the prediction setting, where the future is not available.

At prediction time (right part of Figure 5.2), the prior network is used to generate $\mathbf{z}_{t+1}$ from $\mathbf{y}_t$. Then, $\mathbf{z}_{t+1}$ and $\mathbf{y}_t$ are used as input of the residual dynamics $f_{\boldsymbol{\theta}}$ to predict the next state $\mathbf{y}_{t+1}$. Frame are synthesize with a decoder g_θ that take as input the states $\mathbf{y}_t$, which can be performed in parallel.

5.4 Experimental Setup

This section exposes the experimental results of our method on four standard stochastic video prediction datasets. We compare our method with state-of-the-art baselines on stochastic video prediction. Furthermore, we qualitatively study the dynamics and latent space learned by our model. Training details are described in Appendix C. Animated video samples are available at https://sites.google.com/view/srvp/, and code is available at https://github.com/edouardelasalles/srvp.

5.4.1 Baselines

We compare our model against the following state-of-the-art models:

Stochastic Variational Video Prediction (SV2P)

Babaeizadeh et al. 2018 were the first to use the reparametrization trick and KLD to learn stochastic video generation models. In essence, the generative model is the same as in Bayer et al. 2014: an RNN with independent latent variables added at each timestep (c.f. Section 2.4.2). But they replaced the RNN with ConvLSTMs (X. Shi et al. 2015), more adapted for image processing, and proposed a masking module for handling static backgrounds. The model suffers from the same limitations in terms of dynamics as Bayer et al. 2014.

Stochastic Video Generation (SVG)

E. Denton et al. 2018 improved on that by learning a prior network, similarly to Chung et al. 2015, tying random variables in time. They proposed to use different LSTMs for the recognition module, prior network, and predictor network. They also do not use ConvLSTMs but encode and decode frames in a low dimension vectorial space with deep CNNs. The model obtained better results on a moving robotic arm dataset.

Stochastic Adversarial Video Prediction (SAVP)

Lee et al. 2018 improved SV2P by adding a learned prior module, and an additional Generative Adversarial Model (GAN) (Goodfellow et al. 2014) loss on

the generated images. The added adversarial loss allows them to produce sharper images, while the VAE component maintains diversity in their model.

Structured Variational RNN (StructVRNN)

(Minderer et al. 2019) proposed to extract keypoints from video frames using the unsupervised method of Jakab et al. 2018. They then learn a dynamic variational model on these keypoints using the VRNN from Chung et al. 2015 (c.f. Section 2.4.2). The keypoint extractor and the VRNN dynamics are learned independently.

All baseline results were obtained with pretrained models released by the authors, except for Minderer et al. 2019, where we train their model using their open-source code. Note that we use the same neural architecture as SVG for our encoders and decoders in order to perform fair comparisons with this method, which is the closest to ours among the state-of-the-art. Unless specified otherwise, our model is tested with the same Δt as in training (see Equation 5.2).

5.4.2 Datasets

We present experimental results on a simulated dataset and three real-world datasets:

Stochastic Moving MNIST (SM-MNIST)

This dataset consists of one or two MNIST digits (LeCun et al. 1998) of size 28×28 moving linearly within a 64×64 frame and randomly bounce against its border, sampling a new direction and velocity at each bounce (E. Denton et al. 2018). We use the same settings as E. Denton et al. 2018, train all models on 15 timesteps and condition them at test time on 5 frames. Note that we adapted the dataset to sample more coherent bounces: the original dataset computes digit trajectories that are dependent on the chosen framerate, unlike our corrected version of the dataset. We consequently retrained SVG on this dataset, obtaining comparable results as those originally presented by E. Denton et al. 2018. Test data were produced by generating 5000 samples with a different digit for each sequence coming from the MNIST test set.

KTH Action dataset (KTH)

This dataset is composed of real-world 64×64 videos of 25 people performing one of six actions (walking, jogging, running, boxing, handwaving, and handclapping) in front of different backgrounds (Schüldt et al. 2004). Uncertainty lies in the appearance of subjects, the action they perform, and how it is performed. The

training set is formed with actions from 20 people, the remaining five being used for testing. Training is performed by sampling sub-sequences of size 20 in the train set. The test set is composed of 1000 randomly sampled sub-sequences of size 40 among the videos with the 5 people never seen during training.

Human3.6M

This dataset is also made of videos of subjects performing various actions (Ionescu et al. 2011; Ionescu et al. 2014). While there are more actions and details to capture with less training subjects than in KTH, the video backgrounds are less varied, and subjects always remain within the frames. We use the same settings as Minderer et al. 2019, train all models on 16 timesteps, condition them at test time on 8 frames, and predict 45 frames. Videos used in our experiment are subsampled from the original videos at 6.25Hz, center-cropped from 1000×1000 to 800×800, and resized using the Lanczos of the Pillow library [1] filter to 64×64. Following Minderer et al. 2019, the training set is composed of videos of subjects 1, 5, 6, 7, and 8, and the testing set is made from subjects 9 and 11; videos showing more than one action, marked by "ALL" in the dataset, are excluded. The test set is composed of 1000 randomly sampled sub-sequences of size 40 from the testing videos.

BAIR robot pushing dataset (BAIR)

This dataset contains 64×64 videos of a Sawyer robotic arm pushing objects on a tabletop (Ebert et al. 2017). It is highly stochastic as the arm can change its direction at any moment. Training is performed on 12 frames and testing is done with two conditioning frames on the provided test set, consisting of 256 sequences of 30 frames.

5.4.3 Evaluating Stochastic Predictions

The stochastic nature and novelty of the task of stochastic video prediction make it challenging to evaluate (Lee et al. 2018): since videos and models are stochastic, comparing the ground truth and a predicted video is not adequate. We thus adopt the common approach (E. Denton et al. 2018; Lee et al. 2018) consisting of, for each test sequence, sampling from the tested model a given number (here, 100) of possible futures and reporting the best performing sample against the true video. This method is not perfect but naturally evaluate two aspects of models: their precision and their diversity. If a model has low precision, it will produce blurry samples that will lead to poor scores. If the model is not diverse enough, it will fail to produce the correct sequence since the number of attempts is limited

1. https://pillow.readthedocs.io/

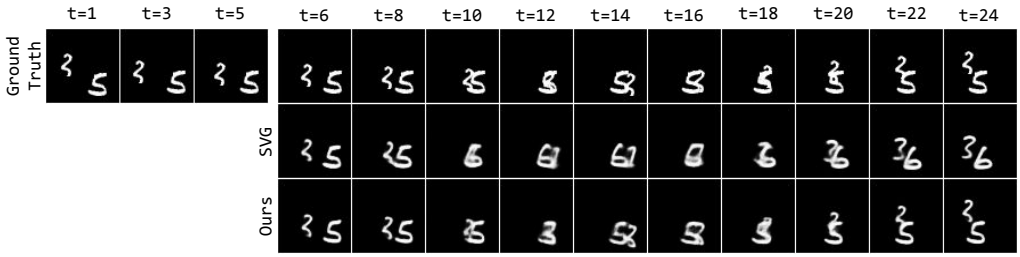
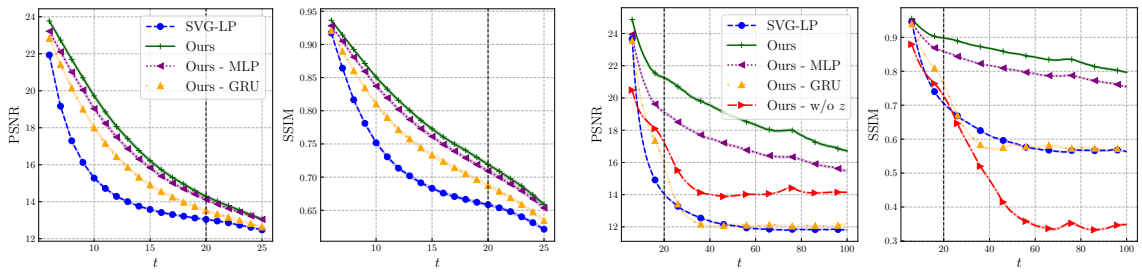


Figure 5.3 – Conditioning frames and corresponding ground truth and best samples with respect to PSNR from SVG and our method for an example of the SM-MNIST dataset.



(a) Stochastic Moving MMNIST.

(b) Deterministic Moving MMNIST.

Figure 5.4 – Mean Peak Signal-to-Noise Ratio (PSNR) and Structured Similarity (SSIM) scores with respect to t for all tested models on the SM-MNIST dataset, with their 95%-confidence intervals. The intervals may be not clearly visible has they are very tight (see Table 5.1). Vertical bars mark the length of train sequences.

(here, 100). Hence to obtain a good score, models have to be sharp and diverse, which are the features we want for a stochastic video prediction model.

We report this discrepancy for three commonly used metrics: PSNR (*higher is better*), SSIM (*higher is better*), and Learned Perceptual Image Patch Similarity (LPIPS) (*lower is better*) (R. Zhang et al. 2018). PSNR tends to promote blurry predictions, as it is a pixel-level measure derived from the ℓ_2 distance, but greatly penalizes errors in predicted positions of objects in the scenes. SSIM is a similarity metric between image patches. LPIPS is a learned distance between activations of deep CNNs trained on image classification tasks, which has been shown to better correlate with human judgment on real images. While these three metrics are computed frame-wise, the recently proposed Fréchet Video Distance (FVD) (*lower is better*) (Unterthiner et al. 2018) aims at directly comparing the distribution of predicted videos with the ground truth distribution through the representations computed by a deep CNN trained on action recognition tasks. It has been shown, independently from LPIPS, to better correlate with human judgment than PSNR and SSIM. We treat all four metrics as complementary, as they capture different

Table 5.1 – Numerical results (mean and 95%-confidence interval) for PSNR and SSIM for tested methods on the two-digits Moving MMNIST dataset. Bold scores indicate the best performing method and, where appropriate, scores whose means lie in the confidence interval of the best performing method.

Models	Stochastic		Deterministic	
	PSNR	SSIM	PSNR	SSIM
SVG	14.45 ± 0.06	0.7070 ± 0.0021	12.93 ± 0.05	0.6245 ± 0.0022
Ours	$\mathbf{16.90 \pm 0.09}$	$\mathbf{0.7789 \pm 0.0025}$	$\mathbf{16.49 \pm 0.06}$	$\mathbf{0.7808 \pm 0.0020}$
Ours - MLP	16.55 ± 0.09	0.7693 ± 0.0024	14.32 ± 0.06	0.6895 ± 0.0023
Ours - GRU	15.81 ± 0.08	0.7463 ± 0.0023	13.16 ± 0.05	0.6318 ± 0.0022

modalities. PSNR challenges the dynamics of the predicted videos, while SSIM rather compares local frame patches but loses some dynamics information. LPIPS and FVD both measure the realism of the predictions compared to the ground truth. FVD considers videos as a whole, making it more capable of detecting temporal inconsistencies. On the other hand, the frame-wise LPIPS metric penalizes more the temporal drifts of videos, since it directly compares each predicted and ground truth frame.

5.5 Results

5.5.1 Prediction

Stochastic Moving MNIST

Figure 5.4a shows quantitative results with two digits. Our model outperforms SVG on both PSNR and SSIM; LPIPS and FVD are not reported as they are not relevant for this synthetic task. Decoupling dynamics from image synthesis allows our method to maintain temporal consistency despite high-uncertainty frames where crossing digits become indistinguishable. For instance in Figure 5.3, the digits shape changes after they cross in the SVG prediction, while our model predicts the correct digits. To evaluate the predictive ability on a longer horizon, we perform experiments on the classic deterministic version of the dataset (Srivastava et al. 2015). We show the results up to $t + 95$ in Figure 5.4b. We can see that our model better captures the dynamics of the problem compared to SVG as its performance decreases significantly less, even at a long-term horizon.

We also compare to two alternative versions of our model in Figure 5.4, where the residual dynamic function is replaced by an MLP or a Gated Recurrent Unit

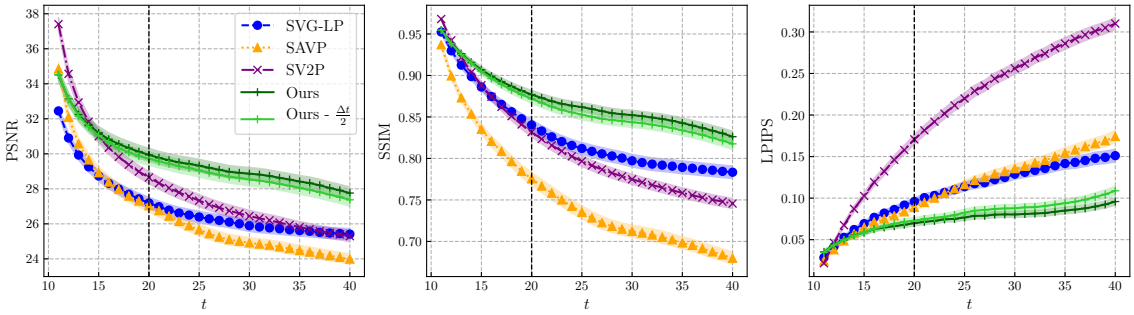


Figure 5.5 – PSNR, SSIM, and LPIPS scores with respect to t for all tested models on the KTH dataset.

Table 5.2 – Numerical results (mean and 95%-confidence interval, when relevant) for PSNR, SSIM, LPIPS, and FVD for tested methods on the KTH dataset. Bold scores indicate the best performing method for each metric and, where appropriate, scores whose means lie in the confidence interval of the best performing method.

Models	PSNR	SSIM	LPIPS	FVD
SV2P	28.18 ± 0.39	0.8141 ± 0.0068	0.2049 ± 0.0080	636 ± 1
SAVP	26.51 ± 0.36	0.7560 ± 0.0083	0.1120 ± 0.0058	374 ± 3
SVG-FP	26.99 ± 0.33	0.8291 ± 0.0074	0.1083 ± 0.0058	377 ± 6
Ours	$\mathbf{29.69 \pm 0.37}$	$\mathbf{0.8697 \pm 0.0057}$	$\mathbf{0.0736 \pm 0.0036}$	$\mathbf{222 \pm 3}$
Ours - GRU	29.13 ± 0.38	0.8590 ± 0.0060	0.0790 ± 0.0039	240 ± 5
Ours - MLP	$\mathbf{29.49 \pm 0.38}$	0.8626 ± 0.0061	0.0825 ± 0.0042	255 ± 4

(GRU) network. Our residual model outperforms both versions on the stochastic, and especially on the deterministic version of the dataset, showing its intrinsic advantage at modeling dynamics. Finally, on the deterministic version of Moving MNIST, we compare to an alternative where auxiliary variables $\mathbf{z}$ are entirely removed, resulting in a temporal model very close to the one presented in Chen et al. 2018. The loss of performance of this alternative model is significant, especially in SSIM, showing that our stochastic residual model offers a substantial advantage even when used in a deterministic environment.

As SV2P and SAVP were not tested on this dataset (in particular, with no pretrain model, code, or hyper-parameters), we only report scores for SVG as the state-of-the-art model on SM-MNIST.

KTH

On this dataset, we substantially outperform on every considered baseline for each metric, as shown in Figure 5.5 and Table 5.2. In some videos, the subject only

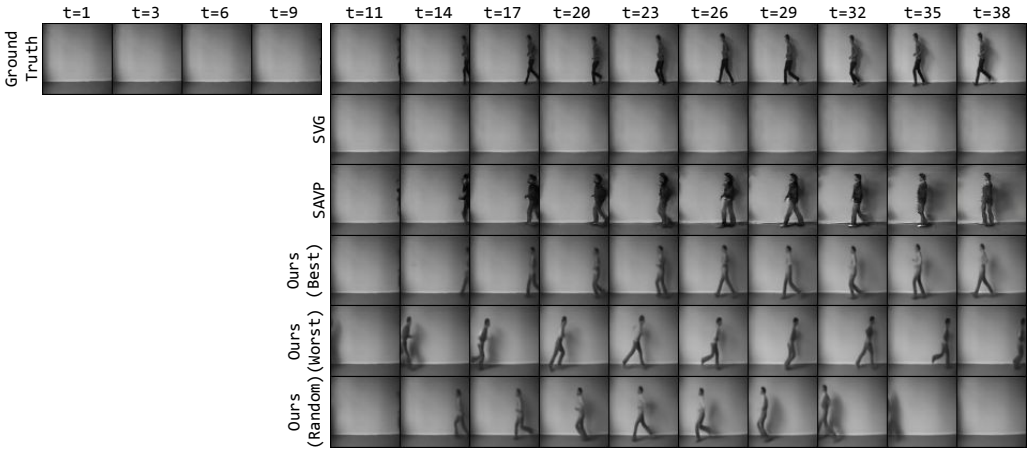


Figure 5.6 – Conditioning frames and corresponding ground truth, best samples from SVG, SAVP and our method, and worst and random samples from our method, for an example of the KTH dataset. Samples are chosen according to their LPIPS with respect to the ground truth. SVG fails to make a person appear unlike SAVP and our model, which better predicts the pose of the subject.

Table 5.3 – FVD scores for SVG and our method on KTH, trained either with DCGAN or VGG encoders and decoders, with their 95%-confidence intervals over five different samples from the models.

Dataset	SVG - VGG	SVG - DCGAN	Ours - VGG	Ours - DCGAN
FVD	377 ± 6	542 ± 6	220 ± 2	371 ± 3

appears after the conditioning frames, requiring the model to sample the moment and location of the subject appearance, as well as its action. This critical case is illustrated in Figure 5.6. There, SVG fails to even generate a moving person; only SAVP and our model manage to do so, and our best sample is closer to the subject's poses compared to SAVP. Moreover, the worst sample of our model demonstrates that it captures the diversity of the dataset by making a person appear at different timesteps and speeds.

To further study the influence of the encoder and decoder architecture on SVG and our model, we train both models with a lighter encoder/decoder architecture. In the presented results, we used the same VGG16 architecture (Simonyan et al. 2015) as SVG. Here, we replace this architecture by the DCGAN architecture (Radford et al. 2016) which has approximately three times fewer layers. The results are presented in Figure 5.7 and Table 5.3.

Since DCGAN is a less powerful architecture than VGG, the results of each method with VGG are expectedly better than those of the same method with

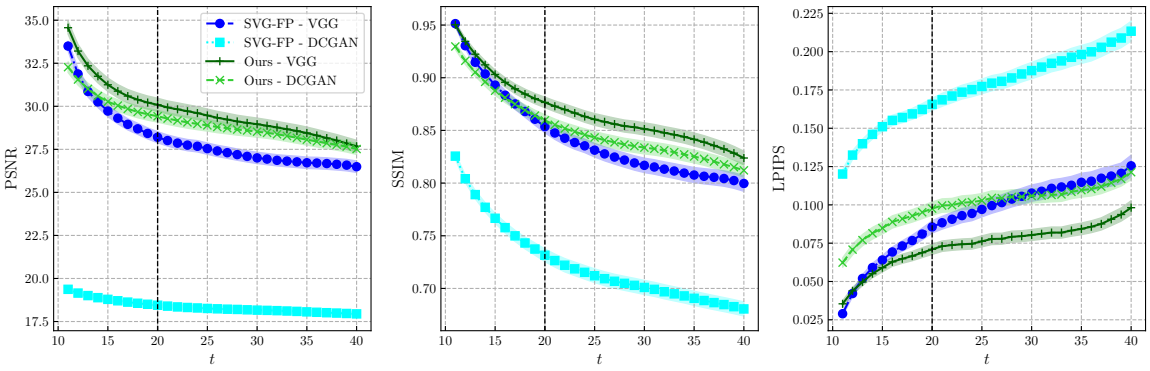


Figure 5.7 – PSNR, SSIM, and LPIPS scores with respect to t on the KTH dataset for SVG and our model with two choices of encoder and decoder architecture for each: DCGAN and VGG.

DCGAN. Moreover, our model outperforms SVG for any fixed choice of encoder and decoder architecture, which is coherent with Figure 5.5.

We observe, however, that the difference between a method using VGG and its DCGAN counterpart differs depending on the model. Our model shows more robustness to the choice of encoder and decoder architecture, as it loses much less performance than SVG when switching to a less powerful architecture. This loss is particularly pronounced with respect to PSNR, which is the metric that penalizes most dynamics errors. This shows that reducing the capacity of the encoders and decoders of SVG not only hurts its ability to produce realistic frames, as expected, but also substantially lowers its ability to learn good dynamics. We assume that this phenomenon is caused by the autoregressive nature of SVG, which makes it dependent on the performance of its encoders and decoders. This supports our motivation to propose a non-autoregressive model for stochastic video prediction.

Finally, Table 5.2 compares our method to its MLP and GRU alternative versions, leading to two conclusions. Firstly, it confirms the structural advantage of residual dynamics observed on Moving MNIST. Indeed, both MLP and GRU lose on all metrics, and especially in terms of realism according to LPIPS and FVD. Secondly, all three versions of our model (residual, MLP, GRU) outperform prior methods. Therefore, this improvement is due to their common inference method, latent nature, and content variable, strengthening again our motivation to propose a non-autoregressive model.

Human3.6M

This dataset is similar to KTH, with more actions and details to capture and less training subjects. However, the video backgrounds are less varied, and subjects always remain within the frames.

As reported in Figure 5.9 and Table 5.4, our model significantly outperform StructVRNN on all metrics, which is the state-of-the-art on this dataset and has

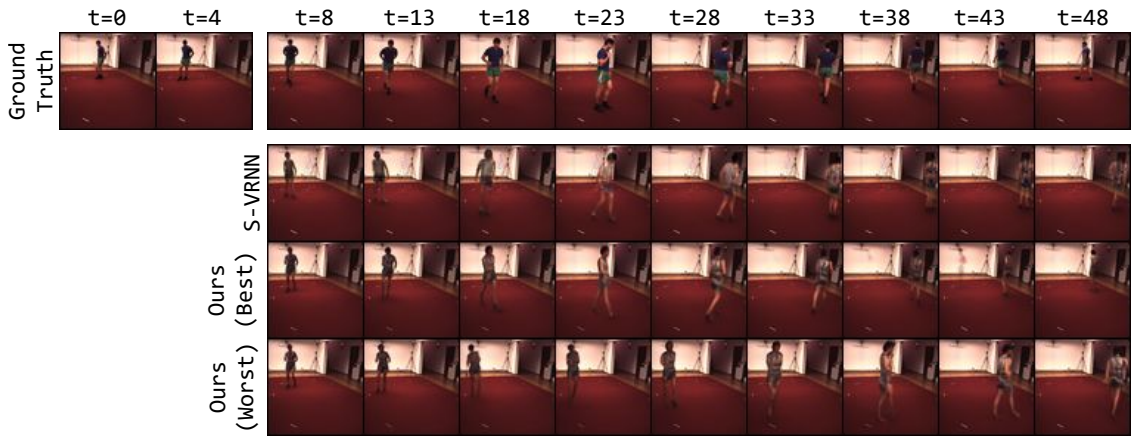


Figure 5.8 – Conditioning frames and corresponding ground truth, best samples from StructVRNN and our method, and worst and random samples from our method, with respect to LPIPS, for a video of the Human3.6M dataset. Our method better captures the dynamic of the subject and produces less artefacts than in StructVRNN predictions.

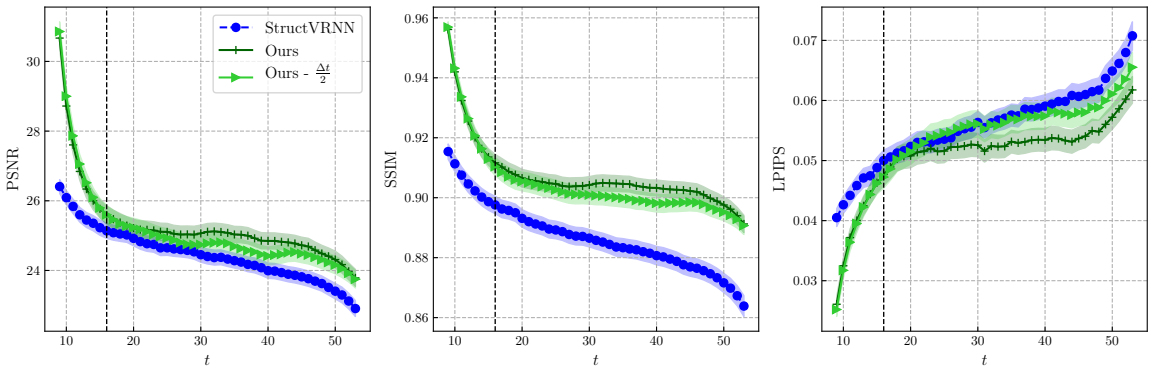


Figure 5.9 – PSNR and LPIPS scores with respect to t for all tested models on the Human3.6M dataset.

Table 5.4 – Numerical results (mean and 95%-confidence interval, when relevant) for PSNR, SSIM, and LPIPS for tested methods on the Human3.6M dataset. Bold scores indicate the best performing method for each metric and, where appropriate, scores whose means lie in the confidence interval of the best performing method.

Models	PSNR	SSIM	LPIPS	FVD
StructVRNN	24.46 ± 0.22	0.8868 ± 0.0031	0.0557 ± 0.0019	556 ± 9
Ours	$\mathbf{25.30 \pm 0.25}$	$\mathbf{0.9074 \pm 0.0028}$	$\mathbf{0.0509 \pm 0.0019}$	416 ± 5
Ours - GRU	23.55 ± 0.26	0.8864 ± 0.0031	0.0691 ± 0.0024	582 ± 4
Ours - MLP	25.00 ± 0.26	0.9047 ± 0.0028	0.0529 ± 0.0019	1050 ± 20

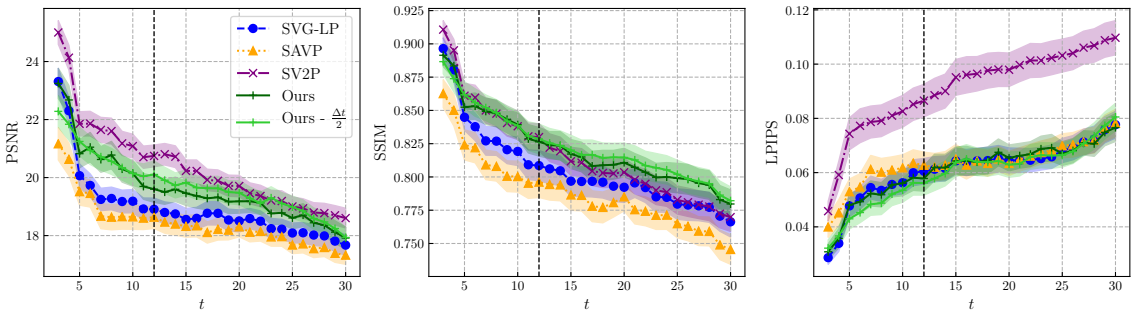


Figure 5.10 – PSNR, SSIM, and LPIPS scores with respect to t for all tested models on the BAIR dataset.

Table 5.5 – Numerical results (mean and 95%-confidence interval, when relevant) with respect to PSNR, SSIM, LPIPS, and FVD for tested methods on the BAIR dataset. Bold scores indicate the best performing method for each metric and, where appropriate, scores whose means lie in the confidence interval of the best performing method.

Models	PSNR	SSIM	LPIPS	FVD
SV2P	**20.39 ± 0.42**	**0.8169 ± 0.0110**	0.0912 ± 0.0063	965 ± 17
SAVP	18.44 ± 0.40	0.7886 ± 0.0117	0.0634 ± 0.0048	**152 ± 9**
SVG	18.95 ± 0.41	0.8057 ± 0.0116	**0.0609 ± 0.0046**	255 ± 4
Ours	19.64 ± 0.45	**0.8211 ± 0.0110**	0.0610 ± 0.0048	198 ± 8

been shown to surpass both SAVP and SVG by Minderer et al. 2019. Figure 5.8 shows the dataset challenges; in particular, both methods do not capture well the subject's appearance. Nonetheless, our model better captures its movements and produces more realistic frames.

Comparisons to the MLP and GRU versions demonstrate once again the advantage of using residual dynamics. GRU obtains very low scores on all metrics, which is coherent with similar results for SVG reported by Minderer et al. 2019. While the MLP version remains close to the residual model on PSNR, LPIPS, and SSIM, it is largely beaten by the latter in terms of FVD.

BAIR

On the BAIR dataset, we achieve similar or better results compared to state-of-the-art models, as Figure 5.10 and Table 5.5 show. We obtain second-best PSNR results behind SV2P, but the latter produces very blurry samples, as can be seen in Figure 5.11, yielding prohibitive LPIPS and FVD scores. In contrast, we achieve the highest SSIM overall, as well as state-of-the-art LPIPS and competitive FVD among these models.

Figure 5.11 – Conditioning frames and corresponding ground truth, best samples from SV2P, SVG, SAVP, and our method, and worst and random samples from our method, with respect to LPIPS, for a video of the BAIR dataset.

5.5.2 Varying Frame Rate in Testing.

We challenge the ability of our model to use a different Euler step size than the one used in training (see Equation 5.2). Figure 5.5 and Figure 5.10 include corresponding results with a halved Δt. Prediction performances remain stable while generating twice as many frames. Our model is thus robust to the refinement of the Euler approximation, showing the quality of the learned dynamics which is close to continuous. This can be used to generate frames at a higher frame rate than the training videos without supervision.

To further study the influence of the Euler step size, we tested our model on varying values of Δt. The results are presented in Table 5.6 for BAIR trained with $\Delta t = 1$. It shows that, when refining the Euler approximation, our model can improve its performance in a setting that is unseen during training. Results stabilize when Δt is small enough, showing that the model is close to the continuous limit.

Table 5.7 and Table 5.8 detail the numerical results of the same experiment on KTH where our model is trained with, respectively, $\Delta t = 1$ and $\Delta t = \frac{1}{2}$, and tested with different values of Δt. They show that if Δt is chosen too high when training (here, $\Delta t = 1$), the model drops in performance when refining the Euler approximation. We assume that this phenomenon arises because the Euler approximation used in training is too rough, making the model adapt to a very discretized dynamic that cannot be transferred to smaller Euler step sizes. When training with smaller step size, (here, $\Delta t = \frac{1}{2}$), results in the training settings are

Table 5.6 – Numerical results for PSNR, SSIM, and LPIPS on BAIR of our model trained with $\Delta t = 1$ and tested with different values of Δt.

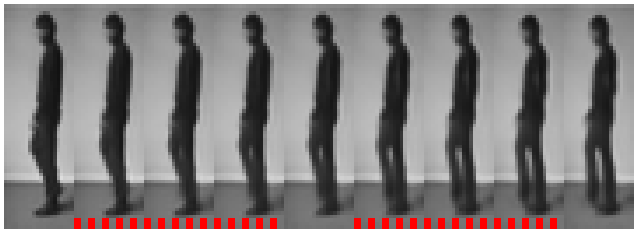
Step size Δt	PSNR	SSIM	LPIPS
$\mathbf{\Delta t = 1}$	19.64 ± 0.45	0.8210 ± 0.0110	0.0612 ± 0.0048
$\Delta t = \frac{1}{2}$	19.76 ± 0.44	0.8235 ± 0.0110	0.0597 ± 0.0047
$\Delta t = \frac{1}{3}$	19.82 ± 0.45	0.8245 ± 0.0111	0.0593 ± 0.0048
$\Delta t = \frac{1}{4}$	19.83 ± 0.46	0.8242 ± 0.0111	0.0593 ± 0.0049
$\Delta t = \frac{1}{5}$	19.85 ± 0.46	0.8243 ± 0.0111	0.0591 ± 0.0048

Table 5.7 – Numerical resuls for PSNR, SSIM, and LPIPS on KTH of our model trained with $\Delta t = 1$ and tested with different values of Δt.

Step size Δt	PSNR	SSIM	LPIPS
$\mathbf{\Delta t = 1}$	29.76 ± 0.38	0.8681 ± 0.0057	0.0737 ± 0.0057
$\Delta t = \frac{1}{2}$	29.05 ± 0.42	0.8539 ± 0.0066	0.0882 ± 0.0050
$\Delta t = \frac{1}{3}$	29.05 ± 0.42	0.8509 ± 0.0069	0.0924 ± 0.0055
$\Delta t = \frac{1}{4}$	28.98 ± 0.42	0.8496 ± 0.0069	0.0939 ± 0.0056
$\Delta t = \frac{1}{5}$	28.95 ± 0.42	0.8490 ± 0.0070	0.0948 ± 0.0057

Table 5.8 – Numerical results for PSNR, SSIM, and LPIPS on KTH of our model trained with $\Delta t = \frac{1}{2}$ and tested with different values of Δt.

Step size Δt	PSNR	SSIM	LPIPS
$\Delta t = 1$	28.80 ± 0.41	0.8495 ± 0.0068	0.0994 ± 0.0057
$\mathbf{\Delta t = \frac{1}{2}}$	29.69 ± 0.37	0.8697 ± 0.0057	0.0736 ± 0.0036
$\Delta t = \frac{1}{3}$	29.52 ± 0.38	0.8656 ± 0.0059	0.0777 ± 0.0041
$\Delta t = \frac{1}{4}$	29.43 ± 0.39	0.8633 ± 0.0061	0.0790 ± 0.0042
$\Delta t = \frac{1}{5}$	29.35 ± 0.39	0.8615 ± 0.0062	0.0810 ± 0.0045



(a) Cropped KTH sample.

(b) Cropped Human3.6M sample.

(c) Cropped BAIR sample.

Figure 5.12 – Generation examples at doubled frame rate, using a halved Δt compared to training. Frames including a bottom red dashed bar are intermediate frames.

equivalent while results obtained with a lower Δt are now much closer, if not equivalent, to the nominal ones.

Note that the loss of performance when using a higher Δt in testing than in training, like in Table 5.8, is expected as it corresponds to loosening the Euler approximation compared to training. However, even in this challenging setting, our model maintains state-of-the-art results, demonstrating the quality of the learned dynamics as it can be discretized more finely if needed at the cost of a reasonable drop in performance.

We also show in Figure 5.12 frames generated at a double and quadruple frame rate on BAIR and KTH. Both figures show smooth intermediate generated frames.

5.5.3 Disentangling Dynamics and Content

Let us show that the proposed model actually separates content from dynamics as discussed in Section 5.3.2. To this end, two sequences $\mathbf{x}^s$ and $\mathbf{x}^t$ are drawn from test sets on Human3.6M and BAIR. While $\mathbf{x}^s$ is used for extracting our content variable $\mathbf{w}^s$, dynamic states $\mathbf{y}^t$ are inferred with our model from $\mathbf{x}^t$. New frame sequences $\widehat{\mathbf{x}}$ are finally generated from the fusion of the content vector and the

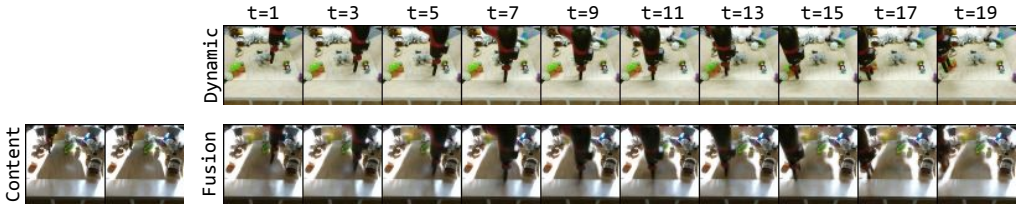


Figure 5.13 – Video (bottom right) generated from the dynamic latent state $\mathbf{y}$ inferred with a video (top) and the content variable $\mathbf{w}$ computed with the conditioning frames of another video (bottom left). The generated video keeps the same background as the bottom left frames, while the robotic arm moves accordingly to the top frames.

dynamics. This results in a content corresponding to the first sequence $\mathbf{x}^s$ while moving according to the dynamics of the second sequence $\mathbf{x}^t$, as observed in Figure 5.13.

5.5.4 Interpolation of Dynamics

Our state-space structure allows us to learn semantic representations in $\mathbf{y}_t$. To highlight this feature, we test whether two Moving MNIST trajectories can be interpolated by linearly interpolating their inferred latent initial conditions. We begin by generating two trajectories $\mathbf{x}^s$ and $\mathbf{x}^t$ of a single moving digit. We infer their respective latent initial conditions $\mathbf{y}_1^s$ and $\mathbf{y}_1^t$. We then use our model to generate frame sequences from latent initial conditions linearly interpolated between $\mathbf{y}_1^s$ and $\mathbf{y}_1^t$. If it learned a meaningful latent space, the resulting trajectories should also be a smooth interpolation between the directions of reference trajectories $\mathbf{x}^s$ and $\mathbf{x}^t$, and this is what we observe in Figure 5.14.

5.6 Conclusion

In this chapter, we tackled the problem of stochasticity in time series prediction. We study the particular case of video prediction, as it has a large impact on other domains, mainly model based reinforcement learning. We introduce a novel dynamic latent model which, unlike prior image-autoregressive models, decouples frame synthesis and dynamics. This temporal model is based on residual updates of a small latent state that is showed to perform better than RNN-based models. We experimentally demonstrate the performance and advantages of the proposed model, which outperforms prior state-of-the-art methods for stochastic video prediction. This work is, to the best of our knowledge, the first to propose a latent dynamic model scaling for video prediction. The proposed model is also novel

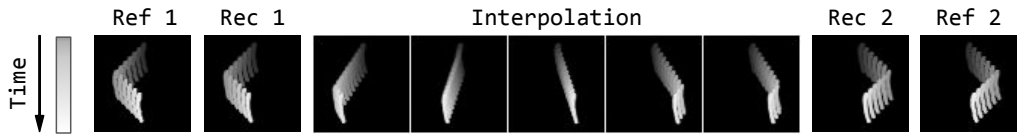


Figure 5.14 – From left to right, $\mathbf{x}^s$, $\widehat{\mathbf{x}}^s$ (reconstruction of $\mathbf{x}^s$ by the VAE of our model), results of the interpolation in the latent space between $\mathbf{x}^s$ and $\mathbf{x}^t$, $\widehat{\mathbf{x}}^t$ and $\mathbf{x}^t$. Each trajectory is materialized in shades of grey in the frames.

with respect to the recent line of work dealing with neural networks and ODEs for temporal modeling; it is the first such residual model to scale to complex stochastic data such as videos.

We believe that the general principles of our model (state-space, residual dynamic, static content variable) can be generally applied to other models as well. Interesting future works include replacing the VRNN model of Minderer et al. 2019 by our dynamics in order to model the evolution of key-points or leveraging the state-space nature of our model in model-based reinforcement learning.

CHAPTER 6

CONCLUSION

Contents

6.1 Summary of Contributions

In this thesis, we tackled different temporal problems with Deep Learning (DL) techniques. The principal idea that guided this thesis was that abstracting temporal components of data in a latent space allows one to learn more efficient dynamic functions for imputation and prediction tasks. Our contributions can be summarized in the following two points.

Latent Dynamics

Throughout all this thesis, the main objective was to capture dynamics of complex phenomena relying principally on observed data and without expert domain knowledge.

We began in Chapter 3 by following recent work on latent temporal models. We inferred latent variables through gradient descent by designing a loss with two objectives: achieving low data reconstruction error while maintaining temporal coherence in the latent space with a dynamic function. Here, the temporal coherence took the form of an ℓ_2 loss between predicted and inferred latent states.

In Chapter 4, we first followed the same design idea as the previous chapter, but in a more principled bayesian framework. By using deep Variational Inference (VI), we were able to infer distributions of latent states, instead of just point estimates. In this model, the temporal alignment between inferred and predicted latent states were optimized via Kullback-Leibler Divergence (KLD) instead of a ℓ_2 loss. Modeling the variance of latent states allows the learning algorithm to find a better equilibrium between data reconstruction and temporal consistency.

However, both models suffer from the total decoupling between the reconstruction loss and the temporal dynamics loss. Indeed, latent variables are inferred independently at each timestep, and the temporal loss is optimized also indepen-

dently for all latent state pairs. In this design, gradients directly backpropagate only between timestep pairs, which makes earning coherent dynamic in the long term difficult, specifically in the presence of multiple trajectories when considering authors. To handle this more complex setting, we fell back to a deterministic dynamic model easier to train.

Finally, in Chapter 5, we extended this deterministic design for stochastic prediction. We once again use deep VI to infer initial states. We took inspiration from recent advances relating residual networks and Ordinary Differential Equations (ODEs), and proposed a stochastic residual function. We incorporated stochastic variables into the deterministic residual functions, allowing us to maintain gradient flow through complete training sequences in a stochastic model. This allowed us to achieve long term stable and diverse predictions.

Structured High-Dimensional Data

In this thesis, we study temporal problems on high-dimensional structured data. In Chapter 3, we explicitly model spatial relations in spatio-temporal data. We model the spatial relations by a weighted graph that represents the distances between time series acquisition locations. Incorporating the adjacency matrix directly in the dynamic function added strong spatial regularization in our model, giving improved prediction and imputation performances compared to black box deep neural network approaches.

In Chapter 4, we focus on textual data. Previously proposed temporal language models focused on diachronic word embeddings learned by some variation of the skip-gram model. However, this algorithm learns word embeddings with limited context, which prevents it from achieving competitive performances on language modeling and downstream tasks, like text classification. We instead proposed to condition a deep recurrent language model by global temporal latent variables

Finally, in Chapter 5, we proposed a video prediction model. We showed that it is possible to completely decouple the image generation process from the dynamics by performing prediction exclusively in a latent space. This allows fast prediction, and high level representation of videos, while keeping visual accuracy of frame samples.

6.2 Perspectives for Future Work

We now discuss some research directions following this work.

Spatial Latent Structure

In our last contribution on video prediction, we abstract all the video dynamics into a single latent vector per timestep. It was possible because the datasets we

used consisted of one subject moving coherently. However, when dealing with more complex videos, relying only on one vector can be limiting.

For instance, in satellite imagery, each pixel represents the value of a physical quantity that follows complex dynamics in time and space. Hence relying on only one vectorial representation means that all the spatial dynamics has to be performed black-boxed by the residual dynamics. This was not a problem on the Stochastic Moving MNIST dataset for instance, since the spatial dynamics can be abstracted to a single point moving linearly. However, on more complex datasets like BAIR, we can already see the limits of the proposed model. Indeed, while the robotic arm is sharp and moves realistically, it is not the case for the surrounding small objects. When the robotic arm pushes such an object, we can see that the model tends to blur the object, and do not manage to handle the more complex spatial dynamics caused by the arm pushing the object (this is visible on the forth "worst" sample on the project website [1]).

So, to enable our model to capture complex spatial dynamics, we could use the graph framework developed in Chapter 3. Combining the latent spatial structure of the STNN model with the powerful dynamics of our video prediction could allow us to tackle such datasets. A first step would be to restrict the scope of study to gridded data like satellite images and more complex videos. That would allow us to rely on convolution networks to perform neighborhood operations. Moreover, scaling operations like downsampling are easy on grids, which allows the reduction of the size of the observation space. Moving forward, recent advances in deep neural networks for graph could help achieve a more general model.

Temporal Latent Structure

In this work, we always considered a unique temporal structure. A direction for future development is to consider hierarchical temporal structures.

Learning hierarchical temporal models can be used to perform temporal segmentation, as in Chung et al. 2017. They use a controller Recurrent Neural Network (RNN) that conditions a RNN Language Model (LM) through Reinforcement Learning (RL). The controller network is not updated at each timestep, but only when the LM outputs a specific token. This model can segment sentences in a paragraph while being trained without supervision. This idea can be used with our video prediction model for instance. On the Stochastic Moving MNIST dataset, the idea would be to learn a controller network able to sample a direction, which would be applied by another network until ideally the digit hits a wall. When that appends, the controller would be asked to sample another valid direction. This framework would segment the temporal dynamics into simple linear movements that are more easily predictable.

1. https://sites.google.com/view/srvp/#h.p_QodL6OGdBAF9

This idea can be pushed further to decompose temporal dynamics into different movement orders. A network is responsible for the first order movement, that is moving the object. A second for the second order, which would indicate to the first network the direction to follow. And a third network for the third order, that would handle acceleration change. In the case of Moving MNIST that would correspond to the rebounds on walls. By decomposing movements this way, the model would learn to temporally segment data when momentum change. Moreover, it could be used to handle several objects together, since the latent temporal dynamics is decomposed into simple operations, and requires fewer latent dimensions.

BIBLIOGRAPHY

Aitchison, Jean (2005). "Language change". In: *The Routledge Companion to Semiotics and Linguistics*, pp. 111–120 (cit. on p. 5).

Ammar, Waleed, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew E. Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni (2018). "Construction of the Literature Graph in Semantic Scholar". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 3 (Industry Papers)*, pp. 84–91 (cit. on p. 64).

Babaeizadeh, Mohammad, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine (2018). "Stochastic Variational Video Prediction". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (cit. on pp. 90–92, 97).

Bahadori, Mohammad Taha, Qi Rose Yu, and Yan Liu (2014). "Fast Multivariate Spatio-temporal Analysis via Low Rank Tensor Learning". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3491–3499 (cit. on p. 34).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (cit. on pp. 19, 20, 24).

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun (2018). "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: *CoRR* abs/1803.01271 (cit. on p. 60).

Bamler, Robert and Stephan Mandt (2017). "Dynamic Word Embeddings". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 380–389 (cit. on pp. 60, 61, 63, 68, 137).

Bańbura, Marta and Michele Modugno (2014). "Maximum likelihood estimation of factor models on datasets with arbitrary pattern of missing data". In: *Journal of Applied Econometrics* 29.1, pp. 133–160 (cit. on p. 9).

Bayer, Justin and Christian Osendorfer (2014). "Learning Stochastic Recurrent Networks". In: *CoRR* abs/1411.7610 (cit. on pp. 30, 97).

Behrmann, Jens, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen (2019). "Invertible Residual Networks". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 573–582 (cit. on p. 96).

Bengio, Samy, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer (2015). "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1171–1179 (cit. on p. 16).

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin (2003). "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3, pp. 1137–1155 (cit. on p. 60).

Bengio, Yoshua, Patrice Y. Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Trans. Neural Networks* 5.2, pp. 157–166 (cit. on p. 14).

Bézenac, Emmanuel de, Arthur Pajot, and Patrick Gallinari (2018). "Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (cit. on p. 26).

Bird, Steven (2006). "NLTK: The Natural Language Toolkit". In: *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006* (cit. on p. 64).

Bishop, C. M., G. E. Hinton, and I. G. D. Strachan (1997). "GTM through time". In: *Fifth International Conference on Artificial Neural Networks (Conf. Publ. No. 440)*, pp. 111–116 (cit. on p. 62).

Blei, David M. and John D. Lafferty (2006). "Dynamic topic models". In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pp. 113–120 (cit. on p. 62).

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3, pp. 993–1022 (cit. on p. 62).

Bowman, Samuel R., Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio (2016). "Generating Sentences from a Continuous Space". In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pp. 10–21 (cit. on pp. 67, 72).

Bradbury, James, Stephen Merity, Caiming Xiong, and Richard Socher (2017). "Quasi-Recurrent Neural Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on p. 17).

Castrejón, Lluìs, Nicolas Ballas, and Aaron C. Courville (2019). "Improved Conditional VRNNs for Video Prediction". In: *CoRR* abs/1904.12165 (cit. on p. 92).

Che, Zhengping, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu (2016). "Recurrent Neural Networks for Multivariate Time Series with Missing Values". In: *CoRR* abs/1606.01865 (cit. on pp. 12, 54).

Chen, Tian Qi, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud (2018). "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 6572–6583 (cit. on pp. 5, 26, 91, 102).

Chiappa, Silvia, Sébastien Racanière, Daan Wierstra, and Shakir Mohamed (2017). "Recurrent Environment Simulators". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on p. 16).

Chiu, Chung-Cheng, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani (2018). "State-of-the-Art Speech Recognition with Sequence-to-Sequence Models". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pp. 4774–4778 (cit. on p. 60).

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734 (cit. on pp. 13, 17, 19).

Chung, Junyoung, Sungjin Ahn, and Yoshua Bengio (2017). "Hierarchical Multiscale Recurrent Neural Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net (cit. on p. 115).

Chung, Junyoung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio (2015). "A Recurrent Latent Variable Model for Sequential Data". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2980–2988 (cit. on pp. 30, 92, 97, 98).

Collins, Jasmine, Jascha Sohl-Dickstein, and David Sussillo (2017). "Capacity and Trainability in Recurrent Neural Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on p. 18).

Connor, Jerome T., R. Douglas Martin, and Les E. Atlas (1994). "Recurrent neural networks and robust time series prediction". In: *IEEE Trans. Neural Networks* 5.2, pp. 240–254 (cit. on p. 13).

Cressie, Noel A. C. and Christopher K. Wikle (2011). *Statistics for spatio-temporal data*. Wiley series in probability and statistics. Hoboken, N.J. Wiley (cit. on p. 34).

Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov (2019). "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, pp. 2978–2988 (cit. on p. 24).

Delasalles, Edouard, Sylvain Lamprier, and Ludovic Denoyer (2019a). "Dynamic Neural Language Models". In: *Neural Information Processing - 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12-15, 2019, Proceedings, Part III*, pp. 282–294 (cit. on pp. 5, 60).

Delasalles, Edouard, Sylvain Lamprier, and Ludovic Denoyer (2019b). "Learning Dynamic Author Representations with Temporal Language Models". In: *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pp. 120–129 (cit. on pp. 5, 60).

Delasalles, Edouard, Ali Ziat, Ludovic Denoyer, and Patrick Gallinari (2019c). "Spatio-temporal neural networks for space-time data modeling and relation discovery". In: *Knowledge and Information Systems* 61.3, pp. 1241–1267 (cit. on pp. 4, 34).

Denton, Emily L. and Vighnesh Birodkar (2017). "Unsupervised Learning of Disentangled Representations from Video". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 4414–4423 (cit. on pp. 92, 94).

Denton, Emily and Rob Fergus (2018). "Stochastic Video Generation with a Learned Prior". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 1182–1191 (cit. on pp. 90–92, 97–99, 141).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186 (cit. on pp. 24, 60, 64, 87).

Ebert, Frederik, Chelsea Finn, Alex X. Lee, and Sergey Levine (2017). "Self-Supervised Visual Planning with Temporal Skip Connections". In: *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, pp. 344–356 (cit. on p. 99).

Eger, Steffen and Alexander Mehler (2016). "On the Linearity of Semantic Change: Investigating Meaning Variation via Dynamic Graph Models". In: *Proceedings*

of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers (cit. on p. 62).

Fan, Hehe, Linchao Zhu, and Yi Yang (2019). "Cubic LSTMs for Video Prediction". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 8263–8270 (cit. on p. 91).

Fedus, William, Ian J. Goodfellow, and Andrew M. Dai (2018). "MaskGAN: Better Text Generation via Filling in the _____". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (cit. on p. 60).

Finn, Chelsea, Ian J. Goodfellow, and Sergey Levine (2016). "Unsupervised Learning for Physical Interaction through Video Prediction". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 64–72 (cit. on pp. 91, 92).

Fraccaro, Marco, Simon Kamronn, Ulrich Paquet, and Ole Winther (2017). "A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 3601–3610 (cit. on p. 143).

Franceschi*, Jean-Yves, Edouard Delasalles*, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari (2020). "Stochastic Latent Residual Video Prediction". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020* (cit. on pp. 6, 90).

Frermann, Lea and Mirella Lapata (2016). "A Bayesian Model of Diachronic Meaning Change". In: *TACL* 4, pp. 31–45 (cit. on p. 62).

Gal, Yarin and Zoubin Ghahramani (2015). "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference". In: *CoRR* abs/1506.02158 (cit. on p. 22).

Gal, Yarin and Zoubin Ghahramani (2016). "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1019–1027 (cit. on p. 22).

Ganeshapillai, Gartheeban, John V. Guttag, and Andrew Lo (2013). "Learning Connections in Financial Time Series". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 109–117 (cit. on p. 45).

Gerrish, Sean and David M. Blei (2010). "A Language-based Approach to Measuring Scholarly Impact". In: *Proceedings of the 27th International Conference on*

Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel, pp. 375–382 (cit. on p. 62).

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2672–2680 (cit. on pp. 16, 91, 97).

Gooijer, Jan G De and Rob J Hyndman (2006). "25 years of time series forecasting". In: *International journal of forecasting* 22.3, pp. 443–473 (cit. on p. 8).

Goyal, Anirudh, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio (2016). "Professor Forcing: A New Algorithm for Training Recurrent Networks". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 4601–4609 (cit. on p. 16).

Grave, Edouard, Armand Joulin, and Nicolas Usunier (2017a). "Improving Neural Language Models with a Continuous Cache". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on p. 23).

Grave, Edouard, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski (2017b). "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pp. 427–431 (cit. on p. 74).

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey E. Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pp. 6645–6649 (cit. on p. 13).

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5-6, pp. 602–610 (cit. on p. 13).

Greff, Klaus, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber (2017). "LSTM: A Search Space Odyssey". In: *IEEE Trans. Neural Netw. Learning Syst.* 28.10, pp. 2222–2232 (cit. on p. 18).

Gregor, Karol, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra (2015). "DRAW: A Recurrent Neural Network For Image Generation". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1462–1471 (cit. on p. 13).

Gregor, Karol, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber (2019). "Temporal Difference Variational Auto-Encoder". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (cit. on p. 90).

Hall, David, Daniel Jurafsky, and Christopher D. Manning (2008). "Studying the History of Ideas Using Topic Models". In: *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 363–371 (cit. on p. 62).

Hamilton, William L., Jure Leskovec, and Dan Jurafsky (2016). "Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers* (cit. on p. 60).

He, Jiawei, Andreas M. Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal (2018). "Probabilistic Video Generation Using Holistic Attribute Control". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, pp. 466–483 (cit. on p. 92).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, pp. 770–778 (cit. on pp. 5, 26).

Higgins, Irina, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017). "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on pp. 29, 142).

Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580 (cit. on p. 22).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on pp. 15, 91).

Honaker, James, Gary King, Matthew Blackwell, et al. (2011). "Amelia II: A program for missing data". In: *Journal of statistical software* 45.7, pp. 1–47 (cit. on p. 54).

Howard, Jeremy and Sebastian Ruder (2018). "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 328–339 (cit. on pp. 19, 60).

Hsu, Wei-Ning, Yu Zhang, and James R. Glass (2017). "Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 1878–1889 (cit. on p. 29).

Inan, Hakan, Khashayar Khosravi, and Richard Socher (2017). "Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on pp. 22, 64).

Ionescu, Catalin, Fuxin Li, and Cristian Sminchisescu (2011). "Latent structured models for human pose estimation". In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pp. 2220–2227 (cit. on p. 99).

Ionescu, Catalin, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu (2014). "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 36.7, pp. 1325–1339 (cit. on p. 99).

Iwata, Tomoharu, Takeshi Yamada, Yasushi Sakurai, and Naonori Ueda (2012). "Sequential Modeling of Topic Dynamics with Multiple Timescales". In: *TKDD* 5.4, 19:1–19:27 (cit. on p. 62).

Jakab, Tomas, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi (2018). "Unsupervised Learning of Object Landmarks through Conditional Image Generation". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 4020–4031 (cit. on p. 98).

Jia, Xu, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool (2016). "Dynamic Filter Networks". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 667–675 (cit. on p. 91).

Kabán, Ata and Mark A. Girolami (2002). "A Dynamic Probabilistic Model to Visualise Topic Evolution in Text Streams". In: *J. Intell. Inf. Syst.* 18.2-3, pp. 107–125 (cit. on pp. 61, 62).

Kalchbrenner, Nal and Phil Blunsom (2013). "Recurrent Continuous Translation Models". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, pp. 1700–1709 (cit. on p. 19).

Kalchbrenner, Nal, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu (2017). "Video Pixel Networks". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1771–1779 (cit. on pp. 34, 91).

Karl, Maximilian, Maximilian Sölch, Justin Bayer, and Patrick van der Smagt (2017). "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data". In: *5th International Conference on Learning Representations,*

ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (cit. on pp. 31, 32, 90, 143).

Kenter, Tom, Melvin Wevers, Pim Huijnen, and Maarten de Rijke (2015). "Ad Hoc Monitoring of Vocabulary Shifts over Time". In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pp. 1191–1200 (cit. on p. 62).

Kim, Yoon, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov (2014). "Temporal Analysis of Language through Neural Language Models". In: *Proceedings of the Workshop on Language Technologies and Computational Social Science ACL 2014, Baltimore, MD, USA, June 26, 2014*, pp. 61–65 (cit. on p. 60).

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (cit. on pp. 64, 141).

Kingma, Diederik P. and Prafulla Dhariwal (2018). "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 10236–10245 (cit. on p. 91).

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (cit. on pp. 28, 29, 67, 138).

Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net (cit. on p. 34).

Koppula, Hema Swetha and Ashutosh Saxena (2013). "Learning Spatio-Temporal Structure from RGB-D Videos for Human Activity Detection and Anticipation". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 792–800 (cit. on p. 34).

Krishnan, Rahul G., Uri Shalit, and David Sontag (2015). "Deep Kalman Filters". In: *CoRR* abs/1511.05121 (cit. on p. 31).

Krishnan, Rahul G., Uri Shalit, and David A. Sontag (2017). "Structured Inference Networks for Nonlinear State Space Models". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 2101–2109 (cit. on pp. 31, 67, 90).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett, Fernando C. N.

Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, pp. 1106–1114 (cit. on p. 3).

Kulkarni, Vivek, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena (2015). "Statistically Significant Detection of Linguistic Change". In: *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pp. 625–635 (cit. on p. 60).

Kumar, Manoj, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma (2019). "VideoFlow: A Flow-Based Generative Model for Video". In: *CoRR* abs/1903.01434 (cit. on p. 91).

Lample, Guillaume, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer (2016). "Neural Architectures for Named Entity Recognition". In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. Ed. by Kevin Knight, Ani Nenkova, and Owen Rambow. The Association for Computational Linguistics, pp. 260–270 (cit. on p. 19).

Le, Hang, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab (2019). "FlauBERT: Unsupervised Language Model Pre-training for French". In: *CoRR* abs/1912.05372 (cit. on p. 24).

Le, Quoc V., Navdeep Jaitly, and Geoffrey E. Hinton (2015). "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units". In: *CoRR* abs/1504.00941 (cit. on p. 17).

Le, Quoc V. and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1188–1196 (cit. on p. 61).

LeCun, Yann, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4, pp. 541–551 (cit. on p. 3).

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on p. 98).

Lee, Alex X., Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine (2018). "Stochastic Adversarial Video Prediction". In: *CoRR* abs/1804.01523 (cit. on pp. 92, 97, 99).

Li, Yingzhen and Stephan Mandt (2018). "Disentangled Sequential Autoencoder". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 5656–5665 (cit. on pp. 92, 94).

Liang, Xiaodan, Lisa Lee, Wei Dai, and Eric P. Xing (2017). "Dual Motion GAN for Future-Flow Embedded Video Prediction". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 1762–1770 (cit. on p. 91).

Liu, Ziwei, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala (2017). "Video Frame Synthesis Using Deep Voxel Flow". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 4473–4481 (cit. on p. 91).

Long, Zichao, Yiping Lu, Xianzhong Ma, and Bin Dong (2018). "PDE-Net: Learning PDEs from Data". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 3214–3222 (cit. on p. 26).

Lotter, William, Gabriel Kreiman, and David D. Cox (2017). "Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on p. 91).

Lu, Chaochao, Michael Hirsch, and Bernhard Schölkopf (2017). "Flexible Spatio-Temporal Networks for Video Prediction". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2137–2145 (cit. on p. 91).

Lu, Yiping, Aoxiao Zhong, Quanzheng Li, and Bin Dong (2018). "Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3282–3291 (cit. on p. 26).

Mathieu, Michaël, Camille Couprie, and Yann LeCun (2016). "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* (cit. on p. 91).

Melis, Gábor, Chris Dyer, and Phil Blunsom (2018). "On the state-of-the-art of Evaluation in Neural Language Models". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (cit. on pp. 21, 60).

Merity, Stephen, Nitish Shirish Keskar, and Richard Socher (2018a). "An Analysis of Neural Language Modeling at Multiple Scales". In: *CoRR* abs/1803.08240 (cit. on p. 60).

Merity, Stephen, Nitish Shirish Keskar, and Richard Socher (2018b). "Regularizing and Optimizing LSTM Language Models". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (cit. on pp. 21, 60, 64).

Merity, Stephen, Caiming Xiong, James Bradbury, and Richard Socher (2017). "Pointer Sentinel Mixture Models". In: *5th International Conference on Learning Representations, ICLR 2017,Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on p. 21).

Mikolov, Tomas, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur (2010). "Recurrent neural network based language model". In: *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pp. 1045–1048 (cit. on pp. 13, 60).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* Pp. 3111–3119 (cit. on pp. 21, 60, 62).

Mikolov, Tomas and Geoffrey Zweig (2012). "Context dependent recurrent neural network language model". In: *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pp. 234–239 (cit. on p. 13).

Minderer, Matthias, Chen Sun, Ruben Villegas, Forrester Cole, Kevin Murphy, and Honglak Lee (2019). "Unsupervised Learning of Object Structure and Dynamics from Videos". In: *CoRR* abs/1906.07889 (cit. on pp. 92, 98, 99, 106, 111).

Mirowski, Piotr W. and Yann LeCun (2009). "Dynamic Factor Graphs for Time Series Modeling". In: *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part II*, pp. 128–143 (cit. on pp. 11, 37, 46, 54).

Montariol, Syrielle and Alexandre Allauzen (2019). "Empirical Study of Diachronic Word Embeddings for Scarce Data". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP 2019, Varna, Bulgaria, September 2-4, 2019*. Ed. by Ruslan Mitkov and Galia Angelova. INCOMA Ltd., pp. 795–803 (cit. on p. 60).

Müller, Klaus-Robert, A. Smola, Gunnar Rätsch, B. Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik (1999). "Using support vector machines for time series prediction". In: *Advances in kernel methods—support vector learning*, pp. 243–254 (cit. on p. 8).

Oord, Aäron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu (2016a). "WaveNet: A Generative Model for Raw Audio". In: *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, p. 125 (cit. on p. 25).

Oord, Aäron van den, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves (2016b). "Conditional Image Generation with

PixelCNN Decoders". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4790–4798 (cit. on p. 91).

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 1310–1318 (cit. on p. 14).

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 2227–2237 (cit. on p. 60).

Radford, Alec, Luke Metz, and Soumith Chintala (2016). "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* (cit. on pp. 103, 141).

Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). "Language Models are Unsupervised Multitask Learners". In: (cit. on pp. 24, 87).

Ranzato, Marc'Aurelio, Arthur Szlam, Joan Bruna, Michaël Mathieu, Ronan Collobert, and Sumit Chopra (2014). "Video (language) modeling: a baseline for generative models of natural videos". In: *CoRR* abs/1412.6604 (cit. on p. 91).

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1278–1286 (cit. on pp. 28, 67).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, pp. 234–241 (cit. on p. 94).

Rosenfeld, Alex and Katrin Erk (2018). "Deep Neural Models of Semantic Shift". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 474–484 (cit. on pp. 63, 68, 137, 138).

Rousseau, François, Lucas Drumetz, and Ronan Fablet (2019). "Residual Networks as Flows of Diffeomorphisms". In: *Journal of Mathematical Imaging and Vision*, pp. 1–11 (cit. on p. 96).

Rubanova, Yulia, Ricky T. Q. Chen, and David Duvenaud (2019). "Latent ODEs for Irregularly-Sampled Time Series". In: *CoRR* abs/1907.03907 (cit. on p. 90).

Rudolph, Maja R. and David M. Blei (2017a). "Dynamic Bernoulli Embeddings for Language Evolution". In: *CoRR* abs/1703.08052 (cit. on p. 63).

Rudolph, Maja R., Francisco J. R. Ruiz, Susan Athey, and David M. Blei (2017b). "Structured Embedding Models for Grouped Data". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 251–261 (cit. on p. 63).

Saito, Kazumi, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda (2009). "Learning Continuous-Time Information Diffusion Model for Social Behavioral Data Analysis". In: *Advances in Machine Learning, First Asian Conference on Machine Learning, ACML 2009, Nanjing, China, November 2-4, 2009. Proceedings*, pp. 322–337 (cit. on pp. 8, 61).

Santoro, Adam, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Tim Lillicrap (2017). "A simple neural network module for relational reasoning". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 4967–4976 (cit. on p. 94).

Schüldt, Christian, Ivan Laptev, and Barbara Caputo (2004). "Recognizing Human Actions: A Local SVM Approach". In: *17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, August 23-26, 2004*, pp. 32–36 (cit. on p. 98).

Semeniuta, Stanislau, Aliaksei Severyn, and Erhardt Barth (2017). "A Hybrid Convolutional Variational Autoencoder for Text Generation". In: *Proceedings of the 2017 Conference on Empirical Methods in NaturalLanguage Processing, EMNLP 2017, Copenhagen, Denmark, September9-11, 2017*, pp. 627–637 (cit. on p. 72).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers* (cit. on p. 87).

Serban, Iulian Vlad, Alexander G. Ororbia II, Joelle Pineau, and Aaron C. Courville (2017). "Piecewise Latent Variables for Neural Variational Text Processing". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 422–432 (cit. on p. 61).

Shang, Jingbo, Yu Zheng, Wenzhu Tong, Eric Chang, and Yong Yu (2014). "Inferring gas consumption and pollution emission of vehicles throughout a city". In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 1027–1036 (cit. on p. 9).

Shen, Dinghan, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin (2018). "Baseline Needs More Love: On Simple Word-Embedding-Based Models and

Associated Pooling Mechanisms". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 440–450 (cit. on p. 74).

Shi, Weiwei, Yongxin Zhu, Philip S. Yu, Tian Huang, Chang Wang, Yishu Mao, and Yufeng Chen (2016). "Temporal Dynamic Matrix Factorization for Missing Data Prediction in Large Scale Coevolving Time Series". In: *IEEE Access* 4, pp. 6719–6732 (cit. on p. 9).

Shi, Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo (2015). "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 802–810 (cit. on pp. 12, 13, 34, 91, 97).

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (cit. on pp. 103, 141).

Song, Fei and W. Bruce Croft (1999). "A General Language Model for Information Retrieval". In: *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA, November 2-6, 1999*, pp. 316–321 (cit. on p. 60).

Song, Yunlong, Min Liu, Shaojie Tang, and XuFei Mao (2012). "Time series matrix factorization prediction of internet traffic matrices". In: *37th Annual IEEE Conference on Local Computer Networks, Clearwater Beach, FL, USA, October 22-25, 2012*, pp. 284–287 (cit. on p. 9).

Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhutdinov (2015). "Unsupervised Learning of Video Representations using LSTMs". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 843–852 (cit. on pp. 12, 34, 91, 101).

Sukhbaatar, Sainbayar, Arthur Szlam, Jason Weston, and Rob Fergus (2015). "End-To-End Memory Networks". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2440–2448 (cit. on p. 23).

Sutskever, Ilya, James Martens, George E. Dahl, and Geoffrey E. Hinton (2013). "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 1139–1147 (cit. on p. 36).

Sutskever, Ilya, James Martens, and Geoffrey E. Hinton (2011). "Generating Text with Recurrent Neural Networks". In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 1017–1024 (cit. on p. 13).

Taieb, Souhaib Ben and Rob J. Hyndman (2014). "Boosting multi-step autoregressive forecasts". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 109–117 (cit. on p. 45).

Tan, Chenhao and Lillian Lee (2015). "All Who Wander: On the Prevalence and Characteristics of Multi-community Engagement". In: *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pp. 1056–1066 (cit. on p. 64).

Unterthiner, Thomas, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly (2018). "Towards Accurate Generative Models of Video: A New Metric & Challenges". In: *CoRR* abs/1812.01717 (cit. on p. 100).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 5998–6008 (cit. on pp. 24, 60).

Villegas, Ruben, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee (2017). "Decomposing Motion and Content for Natural Video Sequence Prediction". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cit. on p. 91).

Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2017). "Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.4, pp. 652–663 (cit. on pp. 13, 60).

Vondrick, Carl and Antonio Torralba (2017). "Generating the Future with Adversarial Transformers". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2992–3000 (cit. on p. 91).

Walker, Jacob, Abhinav Gupta, and Martial Hebert (2015). "Dense Optical Flow Prediction from a Static Image". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 2443–2451 (cit. on p. 91).

Wang, Chong, David M. Blei, and David Heckerman (2012). "Continuous Time Dynamic Topic Models". In: *CoRR* abs/1206.3298 (cit. on p. 62).

Wang, E., J. Silva, R. Willett, and L. Carin (2011). "Dynamic relational topic model for social network analysis with noisy links". In: *2011 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 497–500 (cit. on p. 62).

Wang, Peilu, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao (2015). "Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network". In: *CoRR* abs/1510.06168 (cit. on p. 19).

Wang, Xuerui and Andrew McCallum (2006). "Topics over time: a non-Markov continuous-time model of topical trends". In: *Proceedings of the Twelfth ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006, pp. 424–433 (cit. on pp. 61, 62).

Wang, Yunbo, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S. Yu (2017). "PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 879–888 (cit. on p. 13).

Watter, Manuel, Jost Tobias Springenberg, Joschka Boedecker, and Martin A. Riedmiller (2015). "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2746–2754 (cit. on p. 32).

Weissenborn, Dirk, Oscar Täckström, and Jakob Uszkoreit (2019). "Scaling Autoregressive Video Models". In: *CoRR* abs/1906.02634 (cit. on p. 25).

Wichers, Nevan, Ruben Villegas, Dumitru Erhan, and Honglak Lee (2018). "Hierarchical Long-term Video Prediction without Supervision". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 6033–6041 (cit. on p. 91).

Wierstra, Daan, Alexander Förster, Jan Peters, and Jürgen Schmidhuber (2010). "Recurrent policy gradients". In: *Logic Journal of the IGPL* 18.5, pp. 620–634 (cit. on p. 13).

Williams, Ronald J and David Zipser (1995). "Gradient-based learning algorithms for recurrent". In: *Backpropagation: Theory, architectures, and applications* 433 (cit. on p. 14).

Xu, Jingwei, Bingbing Ni, and Xiaokang Yang (2018). "Video Prediction via Selective Sampling". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 1712–1722 (cit. on p. 91).

Xue, Tianfan, Jiajun Wu, Katherine L. Bouman, and Bill Freeman (2016). "Visual Dynamics: Probabilistic Future Frame Synthesis via Cross Convolutional Networks". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 91–99 (cit. on p. 92).

Yang, Zichao, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick (2017). "Improved Variational Autoencoders for Text Modeling using Dilated Convolutions". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 3881–3890 (cit. on p. 72).

Yao, Zijun, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong (2018). "Dynamic Word Embeddings for Evolving Semantic Discovery". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM*

2018, Marina Del Rey, CA, USA, February 5-9, 2018, pp. 673–681 (cit. on pp. 60–64).

Yazdi, Saeed Varasteh, Ahlame Douzal Chouakria, Patrick Gallinari, and Manuel Moussallam (2018). "Time Warp Invariant Dictionary Learning for Time Series Clustering: Application to Music Data Stream Analysis". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I*. Ed. by Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim. Vol. 11051. Lecture Notes in Computer Science. Springer, pp. 356–372 (cit. on p. 2).

Yuan, Jing, Yu Zheng, Xing Xie, and Guangzhong Sun (2011). "Driving with knowledge from the physical world". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pp. 316–324 (cit. on p. 47).

Yuan, Jing, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang (2010). "T-drive: driving directions based on taxi trajectories". In: *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*, pp. 99–108 (cit. on p. 47).

Zaheer, Manzil, Amr Ahmed, and Alexander J. Smola (2017a). "Latent LSTM Allocation Joint Clustering and Non-linear Dynamic Modeling of Sequential Data". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, pp. 3967–3976 (cit. on p. 61).

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola (2017b). "Deep Sets". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 3391–3401 (cit. on p. 94).

Zhang, Richard, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang (2018). "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 586–595 (cit. on p. 100).

Zhang, Xiang, Junbo Jake Zhao, and Yann LeCun (2015). "Character-level Convolutional Networks for Text Classification". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, pp. 649–657 (cit. on p. 23).

Ziat, Ali, Gabriella Contardo, Nicolas Baskiotis, and Ludovic Denoyer (2016). "Learning Embeddings for Completion and Prediction of Relationnal Multi-

variate Time-Series". In: *24th European Symposium on Artificial Neural Networks, ESANN 2016, Bruges, Belgium, April 27-29, 2016* (cit. on p. 12).

Ziat*, Ali, Edouard Delasalles*, Ludovic Denoyer, and Patrick Gallinari (2017). "Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery". In: *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pp. 705–714 (cit. on pp. 4, 33).

Zilly, Julian G., Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber (2016). "Recurrent Highway Networks". In: *CoRR* abs/1607.03474 (cit. on p. 21).

APPENDIX

a Learning Dynamic Language Models and Author Representations: Deriving Temporal Word Embedding Methods for Recurrent Language Modeling

We detail here how we adapt temporal word embeddings baselines to recurrent language modeling for the experiment in Chapter 4. The baselines are Dynamic Word Embeddings (DWE) Bamler et al. 2017, and DiffTime Rosenfeld et al. 2018. For both methods, we get rid of the context embeddings and only keep word embeddings $\mathbf{U}$.

a.1 Dynamic Word Embeddings

In DWE Bamler et al. 2017, Gaussian word embeddings are learned at each timestep with a temporal diffusion prior:

$$\mathbf{U}_{t+1}|\mathbf{U}_t \sim \mathcal{N}\left(\frac{U_t}{1+\sigma_t^2/\sigma_0^2}, \frac{1}{\sigma_t^{-2}+\sigma_0^{-2}}I\right),$$

where σ_0^2 and σ_t^2 are hyper-parameters of the model.

We derive their skip-gram algorithm for our setting by maximizing the following approximate Evidence Lower Bound (ELBO):

$$\begin{aligned}\mathcal{L}_{\mathcal{DWE}}(\boldsymbol{\theta},\boldsymbol{\phi}) = \sum_{t=1}^{T} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{U}_t)}\left[\log p_{\boldsymbol{\theta}}(\mathbf{X}^t|\mathbf{U}^t)\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{U}_t)}\left[\log \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{U}_{t-1})}\left[p(\mathbf{U}_t|\mathbf{U}_{t-1})\right]\right] \\ - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{U}_t)}\left[\log q_{\boldsymbol{\phi}}(\mathbf{U}_t)\right],\end{aligned} \tag{1}$$

where $p_{\boldsymbol{\theta}}$ is paramatrized by an Long Short-Term Memory (LSTM). $q_{\boldsymbol{\phi}}$ is a variational Gaussian distribution that factorizes as:

$$q_{\boldsymbol{\phi}}(\mathbf{U}) = \prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\mathbf{U}_t),$$

and $\boldsymbol{\phi}$ are its parameters.

To learn this model, we sample a mini-batches $\mathbf{M}$ that contains text coming from different training timesteps. We must hence rescale the ELBO in Equation 1.

We do so by estimating the probability that a given word appears in a particular mini-batch:

$$\begin{aligned}\mathcal{L}_{minibatch}(\boldsymbol{\theta}, \boldsymbol{\phi}) =& \frac{|\mathbf{X}|}{|\mathbf{M}|}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{U}^{\mathbf{M}})}\left[\sum_{\mathbf{x}\in\mathbf{M}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{U}^{\mathbf{M}})\right] \\ &+ \sum_{\mathbf{u}\in\mathbf{U}^{\mathbf{M}}} \frac{1}{(1-(1-\nu_{\mathbf{u}})^{|\mathbf{M}|})} \sum_{t=1}^{T} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{u})}\left[\log \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{u}_{t-1})}\left[p(\mathbf{u}_t|\mathbf{u}_{t-1})\right]\right] \\ &- \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{u}_t)}\left[\log q_{\boldsymbol{\phi}}(\mathbf{u}_t)\right],\end{aligned}$$

where $\mathbf{U}^{\mathbf{M}}$ are the embeddings of words in $\mathbf{M}$, $\nu_{\mathbf{u}}$ is the apparition frequency of term whose embedding is $\mathbf{u}$ in $\mathbf{X}$, and $|\mathbf{X}|$ (respectively $|\mathbf{M}|$) is the number of words in $\mathbf{X}$ ($\mathbf{M}$). In this formulation, gradient computation does not require any approximation, while allowing it to flow through all timesteps.

a.2 DiffTime

The adaptation of the DiffTime baseline Rosenfeld et al. 2018 is straightforward. It learns a non-linear function d that outputs temporal word embeddings:

$$\mathbf{u}_t = d(\mathbf{u}, t; \boldsymbol{\phi})$$

where $\mathbf{u}$ is a learned word embedding, t is a scalar timestep, and $\boldsymbol{\phi}$ are the function's parameters. We refer the reader to the complete paper for more details on the implementation of d.

For recurrent language modeling adaptation, we simply learn jointly the word embeddings $\mathbf{U}$, the parameters $\boldsymbol{\phi}$ of d and the parameters $\boldsymbol{\theta}$ of an LSTM by maximizing the following likelihood:

$$\mathcal{L}_{\mathcal{DT}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{U}) = \prod_{t=1}^{T} \prod_{\mathbf{x}\in\mathbf{X}^t} \prod_{k=1}^{|\mathbf{x}|-1} p_{\boldsymbol{\theta}}(\mathrm{x}_{k+1}|\mathbf{u}_{1:k}^t).$$

b Stochastic Prediction of Videos: ELBO

We develop in this section the computations of the variational lower bound for the stochastic video prediction model presented in Chapter 5.

Using the original variational lower bound of Kingma et al. 2014 in Equation 2:

$$\begin{aligned}&\log p(\mathbf{x}_{1:T} \mid \mathbf{w}) \\ &\quad \geq \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T})\sim q_{Z,Y}} \log p(\mathbf{x}_{1:T} \mid \widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}, \mathbf{w}) - D_{\mathrm{KL}}\big(q_{Z,Y} \;\big\|\; p(\mathbf{y}_{1:T}, \mathbf{z}_{2:T} \mid \mathbf{w})\big)\end{aligned} \tag{2}$$

$$= \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \log p(\mathbf{x}_{1:T} \mid \widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}, rv) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1, \mathbf{z}_{2:T} \mid \mathbf{x}_{1:T}) \parallel p(\mathbf{y}_1, \mathbf{z}_{2:T})\big) \tag{3}$$

$$= \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \widetilde{\mathbf{y}}_t, \mathbf{w}) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1, \mathbf{z}_{2:T} \mid \mathbf{x}_{1:T}) \parallel p(\mathbf{y}_1, \mathbf{z}_{2:T})\big), \tag{4}$$

where:

- Equation 3 is given by the forward and inference models factorizing p and q in Equation 5.3 and Equation 5.4 and illustrated by, respectively, Figure 5.1a and Figure 5.1b:
  - the $\mathbf{z}$ variables and $\mathbf{y}_1$ are independent from $\mathbf{w}$ with respect to p and q;
  - the $\mathbf{y}_{2:T}$ variables are deterministic functions of $\mathbf{y}_1$ and $\mathbf{z}_{2:T}$ with respect to p and q;
- Equation 4 results from the factorization of $p(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{w})$ in Equation 5.3.

From there, by using the integral formulation of D_{KL}:

$$\begin{aligned}
&\log p(\mathbf{x}_{1:T} \mid \mathbf{w}) \\
&\quad \geq \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \widetilde{\mathbf{y}}_t, \mathbf{w}) \\
&\qquad + \int \cdots \int_{\mathbf{y}_1, \mathbf{z}_{2:T}} q(\mathbf{y}_1, \mathbf{z}_{2:T} \mid \mathbf{x}_{1:T}) \log \frac{p(\mathbf{y}_1, \mathbf{z}_{2:T})}{q(\mathbf{y}_1, \mathbf{z}_{2:T} \mid \mathbf{x}_{1:T})} \, \mathrm{d}\mathbf{z}_{2:T} \, \mathrm{d}\mathbf{y}_1
\end{aligned} \tag{5}$$

$$\begin{aligned}
&= \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \widetilde{\mathbf{y}}_t, \mathbf{w}) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1 \mid \mathbf{x}_{1:T}) \parallel p(\mathbf{y}_1)\big) \\
&\quad + \mathbb{E}_{\widetilde{\mathbf{y}}_1 \sim q(\mathbf{y}_1 \mid \mathbf{x}_{1:T})} \left[\int \cdots \int_{\mathbf{z}_{2:T}} q(\mathbf{z}_{2:T} \mid \mathbf{x}_{1:T}, \widetilde{\mathbf{y}}_1) \log \frac{p(\mathbf{z}_{2:T} \mid \widetilde{\mathbf{y}}_1)}{q(\mathbf{z}_{2:T} \mid \mathbf{x}_{1:T}, \widetilde{\mathbf{y}}_1)} \, \mathrm{d}\mathbf{z}_{2:T} \right]
\end{aligned} \tag{6}$$

$$\begin{aligned}
&= \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \widetilde{\mathbf{y}}_t, \mathbf{w}) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1 \mid \mathbf{x}_{1:k}) \parallel p(\mathbf{y}_1)\big) \\
&\quad + \mathbb{E}_{\widetilde{\mathbf{y}}_1 \sim q(\mathbf{y}_1 \mid \mathbf{x}_{1:k})} \left[\int \cdots \int_{\mathbf{z}_{2:T}} q(\mathbf{z}_{2:T} \mid \mathbf{x}_{1:T}, \widetilde{\mathbf{y}}_1) \log \frac{p(\mathbf{z}_{2:T} \mid \widetilde{\mathbf{y}}_1)}{q(\mathbf{z}_{2:T} \mid \mathbf{x}_{1:T}, \widetilde{\mathbf{y}}_1)} \, \mathrm{d}\mathbf{z}_{2:T} \right]
\end{aligned} \tag{7}$$

$$
\begin{aligned}
&= \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \widetilde{\mathbf{y}}_t, \mathbf{w}) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1 \mid \mathbf{x}_{1:k}) \,\big\|\, p(\mathbf{y}_1)\big) \\
&\quad + \mathbb{E}_{\widetilde{\mathbf{y}}_1 \sim q(\mathbf{y}_1 \mid \mathbf{x}_{1:k})} \left[\int \cdots \int_{\mathbf{z}_{2:T}} \prod_{t=2}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \sum_{t=2}^{T} \log \frac{p(\mathbf{z}_t \mid \widetilde{\mathbf{y}}_1, \mathbf{z}_{2:t-1})}{q(\mathbf{z}_t \mid \mathbf{x}_{1:t})} \, \mathrm{d}\mathbf{z}_{2:T} \right]
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
&= \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \widetilde{\mathbf{y}}_t, \mathbf{w}) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1 \mid \mathbf{x}_{1:k}) \,\big\|\, p(\mathbf{y}_1)\big) \\
&\quad - \mathbb{E}_{\widetilde{\mathbf{y}}_1 \sim q(\mathbf{y}_1 \mid \mathbf{x}_{1:k})} D_{\mathrm{KL}}\big(q(\mathbf{z}_2 \mid \mathbf{x}_{1:t}) \,\big\|\, p(\mathbf{z}_2 \mid \widetilde{\mathbf{y}}_1)\big) \\
&\quad + \mathbb{E}_{\widetilde{\mathbf{y}}_1 \sim q(\mathbf{y}_1 \mid \mathbf{x}_{1:k})} \mathbb{E}_{\widetilde{\mathbf{z}}_2 \sim q(\mathbf{z}_2 \mid \mathbf{x}_{1:2})} \\
&\quad \left[\int \cdots \int_{\mathbf{z}_{3:T}} \prod_{t=3}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \sum_{t=3}^{T} \log \frac{p(\mathbf{z}_t \mid \mathbf{y}_1, \widetilde{\mathbf{z}}_{2:t-1})}{q(\mathbf{z}_t \mid \mathbf{x}_{1:t})} \, \mathrm{d}\mathbf{z}_{3:T} \right],
\end{aligned} \tag{9}
$$

where:

- Equation 7 follows from the inference model of Equation 5.4, where $\mathbf{y}_1$ only depends on $\mathbf{x}_{1:k}$;
- Equation 8 is obtained from the factorizations of Equation 5.3 and Equation 5.4.

By iterating Equation 9's step on $\mathbf{z}_3, \ldots, rv_T$ and factorizing all expectations, we obtain:

$$
\begin{aligned}
&\log p(\mathbf{x}_{1:T} \mid \mathbf{w}) \\
&\quad \geq \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \widetilde{\mathbf{y}}_t, \mathbf{w}) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1 \mid \mathbf{x}_{1:k}) \,\big\|\, p(\mathbf{y}_1)\big) \\
&\qquad - \mathbb{E}_{\widetilde{\mathbf{y}}_1 \sim q(\mathbf{y}_1 \mid \mathbf{x}_c)} \Big(\mathbb{E}_{\widetilde{\mathbf{z}}_t \sim q(\mathbf{z}_t \mid \mathbf{x}_{1:t})} \Big)_{t=2}^{T} \sum_{t=2}^{T} D_{\mathrm{KL}}\big(q(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \,\big\|\, p(\mathbf{z}_t \mid \widetilde{\mathbf{y}}_1, \widetilde{\mathbf{z}}_{1:t-1})\big),
\end{aligned} \tag{10}
$$

and we finally retrieve Equation 5.5 by using the factorization of Equation 5.4:

$$
\begin{aligned}
&\log p(\mathbf{x}_{1:T} \mid \mathbf{w}) \\
&\quad \geq \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \widetilde{\mathbf{y}}_t, \mathbf{w}) - D_{\mathrm{KL}}\big(q(\mathbf{y}_1 \mid \mathbf{x}_{1:k}) \,\big\|\, p(\mathbf{y}_1)\big) \\
&\qquad - \mathbb{E}_{(\widetilde{\mathbf{z}}_{2:T}, \widetilde{\mathbf{y}}_{1:T}) \sim q_{Z,Y}} \sum_{t=2}^{T} D_{\mathrm{KL}}\big(q(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \,\big\|\, p(\mathbf{z}_t \mid \widetilde{\mathbf{y}}_{t-1})\big).
\end{aligned} \tag{11}
$$

c Stochastic Prediction of Videos: Training Details

c.1 Architecture

Encoder and decoder architecture. Both g_θ and h_ϕ are chosen to have different architectures depending on the dataset. We used the same architectures as in E. Denton et al. 2018: a DCGAN discriminator and generator architecture (Radford et al. 2016) for Moving MNIST, and a VGG16 (Simonyan et al. 2015) architecture (mirrored for h_ϕ) for BAIR and KTH. In both cases, the output of h_ϕ (i.e., $\widetilde{\mathbf{x}}$) is a vector of size 128, and g_θ and h_ϕ weights are initialized using a centered normal distribution with a standard deviation of 0.02.

For the Moving MNIST dataset, the content variable $\mathbf{w}$ is obtained directly from $\widetilde{\mathbf{x}}$ and is thus a vector of size 128. For KTH, Human3.6M, and BAIR, we supplement this vectorial variable with skip connections from all layers of the encoder g_θ that are then fed to the decoder h_ϕ to handle complex backgrounds. For Moving MNIST, the number of frames k used to compute the content variable is 5; for KTH and Human3.6M, it is 3; for BAIR, it is 2.

LSTM architecture. The LSTM used for all datasets has a single layer of LSTM cells with a hidden state size of 256.

MLP architecture. All Multi-Layer Perceptrons (MLPs) used in inference (with parameters ϕ) have three linear layers with hidden size 256 and ReLU activations. All MLPs used in the forward model (with parameters θ) have four linear layers with hidden size 512 and ReLU activations. Weights of f_θ, in particular, are orthogonally initialized with a gain of 1.2 for KTH and Human3.6M, and 1.41 for the other datasets, while the other MLPs are initialized with default weight initialization of PyTorch.

Sizes of latent variables. The sizes of the latent variables in our model are the following: for Moving MNIST, $\mathbf{y}$ and $\mathbf{z}$ have size 20; for KTH, Human3.6M, and BAIR, $\mathbf{y}$ and $\mathbf{z}$ have size 50.

Euler step size Models were trained with $\Delta t = 1$ on SM-MNIST, and with $\Delta t = \frac{1}{2}$ on the others datasets.

c.2 Optimization

Loss function. All models are trained using the Adam optimizer (Kingma et al. 2015) with learning rate 3×10^{-4}, and decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The

batch size is chosen to be 128 for Moving MNIST, 100 for KTH and Human3.6M, and 192 for BAIR. The regularization coefficient λ is always set to 1.

For the Moving MNIST dataset, we follow Higgins et al. 2017, and weight the KL divergence terms on $\mathbf{z}$ (i.e., the sum of KL divergences in Equation 5.5) by multiplying them by a factor $\beta = 2$.

Variance of the observation. The variance ν used in the observation probability distribution $\mathcal{G}\big(g_\theta(\mathbf{y})\big) = \mathcal{N}\big(g_\theta(\mathbf{y}), \nu I\big)$ is chosen as follows:

- for Moving MNIST, $\nu = 1$;
- for KTH and Human3.6M, $\nu = 4 \times 10^{-2}$;
- for BAIR, $\nu = \frac{1}{2}$.

Number of optimization steps. The number of optimization steps is the following for the different datasets:

- Moving MNIST (stochastic): 1 000 000 steps, with additional 100 000 steps where the learning rate is linearly decreased to 0;
- Moving MNIST (deterministic): 800 000 steps, with additional 100 000 steps where the learning rate is linearly decreased to 0;
- KTH: 150 000 steps, with additional 50 000 steps where the learning rate is linearly decreased to 0, the final model being chosen among several checkpoints as the one having the best evaluation PSNR (which differs from the test score as we extract from the train set an evaluation set);
- Human3.6M: 325 000 steps, with additional 25 000 steps where the learning rate is linearly decreased to 0, the final model being chosen in the manner as KTH;
- BAIR: 1 000 000 steps, with additional 500 000 steps where the learning rate is linearly decreased to 0.

The evaluation sets of KTH and Human3.6M are chosen by randomly selecting 5% of the training videos from the training set.

d Stochastic Prediction of Videos: Pendulum Experiments

The Pendulum experiment is an addition to Chapter 5. We test the ability of our stochastic video prediction model to model the dynamics of a common

Table 1 – ELBO score for DVBF, KVAE and our model on the Pendulum dataset. The bold score indicates the best performing method.

Score	DVBF	KVAE	Ours
ELBO	798.56	**807.02**	806.12

dataset used in the literature of state-space models (Karl et al. 2017; Fraccaro et al. 2017), Pendulum (Karl et al. 2017). It consists of noisy observations of a dynamic torque-controlled pendulum; it is stochastic as the information of this control is not available. We test our model, without the content variable $\mathbf{w}$, in the same setting as DVBF (Karl et al. 2017) and KVAE (Fraccaro et al. 2017) and report the corresponding ELBO scores in Table 1 The encoders and decoders for all methods are MLPs.

Our model outperforms DVBF and is merely beaten by KVAE. This can be explained by the nature of the KVAE model, whose sequential model is not learned using a Variational Auto-Encoder (VAE) but a Kalman filter allowing exact inference in the latent space. On the contrary, DVBF is learned, like our model, by a sequential VAE, and is thus much closer to our model than KVAE. This result then shows that the dynamic model that we chose in the context of sequential VAEs is more adapted on this dataset than the one of DVBF, and achieve results close to a method taking advantage of exact inference using adapted tools such as Kalman filters.