Neural Time Series Forecasting With Latent Dynamics

Zexuan Yin

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

September 19, 2024

I, Zexuan Yin, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This thesis investigates the use of neural network models for time series forecasting with an emphasis on modelling latent dynamics (unobservable time series).

Time series forecasting is of great research interest to both industry and academia. It describes the task of predicting the future values of one or more time series conditioned on past information. With growth in the availability of data, researchers have started developing machine learning - more specifically neural networks - to model more complex temporal dynamics. Neural networks excel at modelling non-linearity and requires fewer assumptions on the underlying process.

To contribute to the area of deep learning for time series forecasting, I focus specifically on the fact that not all time series can be observed. Financial markets for example, contain unobservable market regimes which drive observed time series such as stock returns and prices. Being able to model these latent dynamics provide us with a useful tool to study what is happening beneath the surface. To achieve this, I bring topics together from other areas of machine learning such as representation learning and Bayesian inference.

This thesis is broken down into four experiments. In the first experiment, I develop a stochastic variant of the recurrent neural network which can be used to perform multi-step-ahead time series forecasting, and generate confidence intervals for the predictions.

In the second experiment, I study the concept of Granger causality in the presence of a potential confounder. I develop a neural network architecture to model the confounder and show that by taking this into account, one can obtain better forecasting accuracy on the target time series.

The third and fourth experiments are concerned with the application of latent variable modelling in financial markets. In experiment three, I bring together deep learning and GARCH models from the field of econometrics and propose a neural architecture for volatility (variance/covariance) forecasting in a low dimensional setting ($<$5 assets). Finally, in the fourth experiment, I build on my work in the third experiment to propose a model capable of forecasting the volatility of an investment portfolio in higher dimensional settings.

Impact Statement

The findings and models proposed in this thesis have many potential applications in both industry and academia.

The stochastic recurrent neural network proposed in experiment one is a generative model for time series capable of generating a distribution of future states conditioned on past information. This model is currently being adopted by the AI Research team at JP Morgan investment bank to study limit order book dynamics and market micro-structure in global equity markets. This brings potential value in the electronic trading business as the bank can use this model to output a distribution of future order book states and test their strategies for optimal order placement, potentially with reinforcement learning. In addition to this, several hedge fund managers have confirmed that this architecture could potentially be used to develop systematic trading strategies.

Granger causality is a commonly used concept in fields such as neuroscience and finance to study the predictability of a target time series conditioned on other time series. However, only observed times series are being used as predictors, and the reliability of the test suffers in the presence of a latent confounder. In experiment two, I demonstrate that by directly accounting for confounding, one can obtain better forecasting of the target time series than using observed time series alone. This finding could help industry professionals and academic researchers to better understand potentially non-linear relationships between different time series such as neural activities inside the brain, and the effect of macroeconomic variables on the stock market.

In financial markets, the concept of volatility is of critical importance as it is closely linked to risk management practices. When developing a trading strategy, the risk adjusted return is a key parameter to consider after taking volatility into account. Since volatility is an unobserved variable, it is often inferred and predicted using observed time series such as stock returns. The volatility models developed in experiments three and four can accommodate more complex relationships between historical returns and risk. The covariance matrix predictions made by the model can be used by commodity traders to compute the Value-at-risk (VAR) of their positions, and asset managers can use these predictions to perform dynamic portfolio optimisation.

Acknowledgements

I would like to express my deepest gratitude towards my principle supervisor Prof Paolo Barucca for his utmost support and patience throughout my PhD journey. It is through his mentorship that I have acquired the research and problem solving skills which will better prepare me for my future career.

I would like to thank my secondary supervisor Prof Fabio Caccioli for always being there for me whenever I needed advice on improving my work. The feedback I gained from him has contributed greatly to my research confidence and work quality.

I am grateful to my examiners: Prof Tomaso Aste and Prof Joerg Osterrieder for taking the time to review my work and provide constructive feedback. It is with their help that I am able to finish my PhD journey on a high note.

I would also like to acknowledge Prof Brooks Paige and Prof Denise Gorse for giving me invaluable advice on my research and providing supervision during my previous vivas.

Last but not least, I thank my family for their continuous support throughout the years and for being my source of motivation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Time series forecasting is widely performed in many industries to predict the future behaviour of variables of interest given historical information. In finance for example, commercial banks use predictions of macroeconomic variables such as unemployment rates and house prices to assess their potential impact on their loan portfolios. In the power industry, since electricity cannot be easily stored, producers attempt to forecast short-term demand and supply to better balance their portfolios. Given the practical importance of time series forecasting, it is not surprising that literature within this field is abundant and studies have focused on various subdomains such as point forecasting, probabilistic forecasting, latent variable modelling, and multistep forecasting etc.

In recent years, time series forecasting with deep neural networks has gained popularity due to easier access to big data and advanced computational power. Neural networks are black-box function approximators capable of modelling non-linear dynamics. Since most real world time series are complex, deep learning provides an alternative way to analyse time series that requires fewer assumptions on the various components of the underlying time series such as linearity, trend, and seasonality. This approach is especially useful when one has little understanding of the process being modelled and has hence decided to adopt a fully data-driven approach to forecasting. The application of neural networks has achieved great success in areas such as computational hydrodynamics, option pricing, astrophysics, and macroeconomics.

Thusfar, the majority of the methods rely on using historical time series as inputs to a neural network and outputting the prediction value(s) of the target time series. This approach works well when all time series values can be observed and acquired. In reality however, there are quantities of interest that are not directly observable (latent) such as market regimes, market volatility, and common factors that influence multiple time series.

How to forecast a time series that is unobserved? Latent variable time series models are a feasible solution. These models consist of a transition dynamics for the latent states, and an emission dynamics to map from the latent space to the observation space. Given the recent success of neural networks in forecasting, it is of interest to explore the intersection of deep learning and latent variable modelling.

In this thesis I attempt to address the following research questions: 1. can deep learning and latent variable modelling be combined to achieve superior forecasting performance, 2. can deep learning be used to infer relationships between latent and observed time series, 3. how to successfully apply this framework to model real world time series.

1.1 Research objectives

For the rest of this thesis, I refer to the time series being forecast as the ”target”, and the input time series used to forecast the target as ”predictors”.

The main aim of this thesis is to leverage the modelling capabilities of neural networks to propose state-of-the-art architectures to analyse latent time series conditioned on observational time series. This then leads to two directions depending on whether the target time series to be forecast is latent or observational. If the target is latent, I propose models to predict future values of the latent process using the observational as input. On the other hand, if the target is observational, I propose a methodology to project the input times series to a latent space and investigate whether learning the shared factors can help to achieve better forecasting on the target time series .

Four experiments have been carried out to address the above objectives. For direction 1 (target is latent): in experiments three and four, I propose neural network architectures to forecast the volatility (covariance matrix) of an investment portfolio. For direction 2 (observational target): in experiment one I demonstrate that learning the shared factors of the predictors can improve forecasting performance; in experiment two, I investigate Granger causality in the presence of a latent confounder.

1.2 Scientific contributions

This thesis advances existing research in the following areas:

- Time series generative modelling. I propose a neural network architecture to learn a structured latent space that enables multistep time series forecasting with confidence intervals. In addition to forecasting, this model can also be used to generate synthetic time series for tasks such as backtesting trading strategies. This methodology has been adopted by an investment bank and has also attracted interests from various hedge funds.

- Granger causality in the presence of potential confounders. I propose a neural network based Granger causality test capable of handling nonlinear temporal dynamics and potential confounding effects between the target and the predictors.

- Volatility forecasting. I design methodologies to predict the covariance matrix and returns distribution of an investment portfolio that are computationally efficient and outperform existing methods used in industry as well as recent machine learning based methods.

- The contents of chapter 3 was published as Zexuan Yin and Paolo Barucca. Stochastic recurrent neural network for multistep time series forecasting. In *Neural Information Processing*, pages 14–26, Cham, 2021. Springer International Publishing. ISBN 978-3-030-92185-9

- The contents of chapter 4 was published as Zexuan Yin and Paolo Barucca. Deep recurrent modelling of granger causality with latent confounding. *Ex-*

pert Systems with Applications, 207, 11 2022. ISSN 09574174. doi: 10.1016/j.eswa.2022.118036

- The contents of chapters 5 and 6 are currently under review

1.3 Thesis outline

The rest of the thesis is structured as follows:

- Chapter 2 presents relevant literature that is used as the foundation of the new methodologies being introduced in subsequent chapters. It contains: a summary of the popular neural network models currently being used for time series forecasting and discusses their strengths and weaknesses, existing methodologies used in the field of volatility forecasting and causal inference, statistical concepts used to develop novel architectures, and evaluation metrics used to assess the effectiveness of different approaches.

- Chapter 3 introduces a novel architecture for multistep time series forecasting designed using a recurrent neural network and the concept of variance inference. It demonstrates the advantages of learning shared temporal dynamics of the covariates on the predictive performance of the target time series. The new methodology is applied to forecast real world data from various fields to showcase its performance. Potential applications of this framework in volatility forecasting and Granger causality analysis are explored in chapters 4, 5 and 6.

- Chapter 4 introduces a methodology for Granger causality analysis in the presence of non-linear relationships and potential confounders. The new methodology shows that directly accounting for confounders results in improved prediction performance compared to existing methods which assume the absence of confounding.

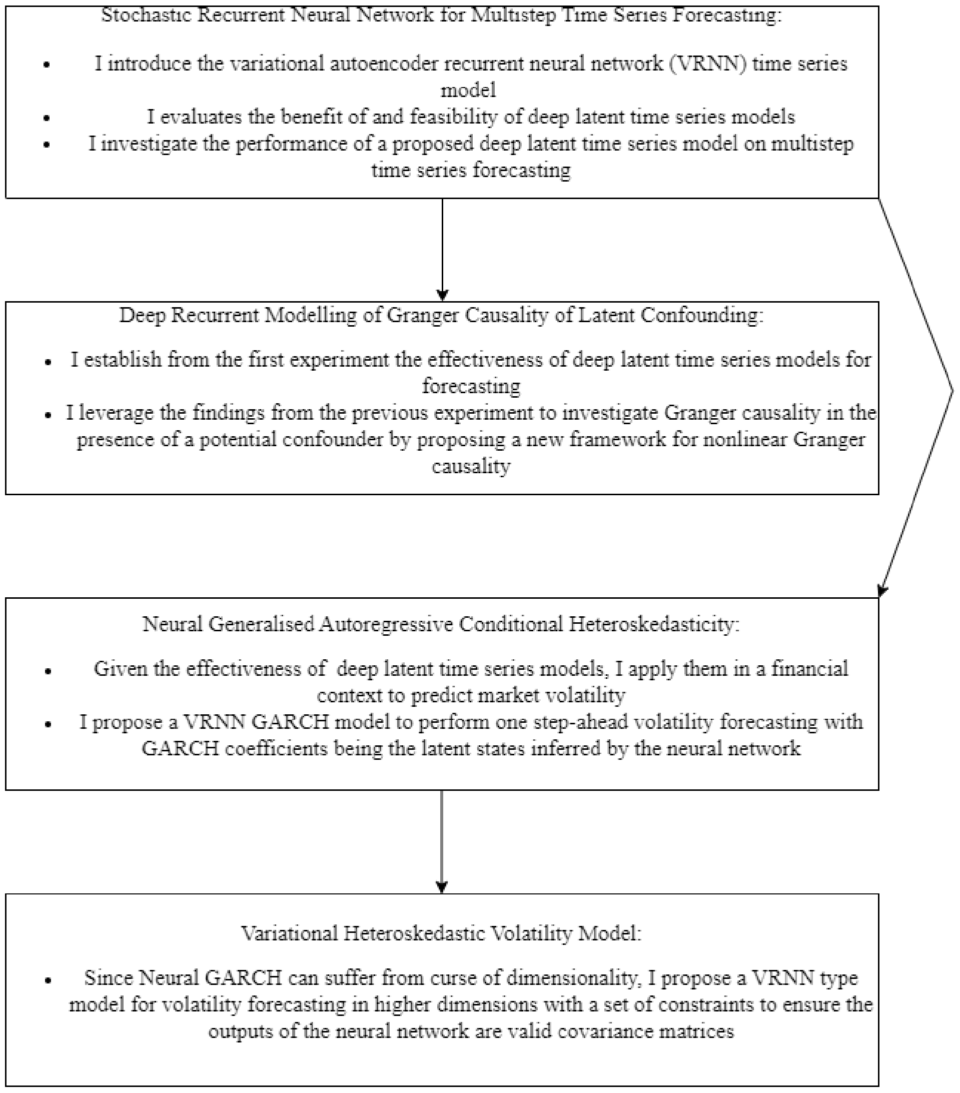
- Chapter 5 proposes a novel methodology to forecast the volatility (covariance matrix) of a small investment portfolio (< 5 assets). This chapter serves as a proof-of-concept to demonstrate the effectiveness of deep learning for volatility forecasting. It uses Generalised Autoregressive Conditional Heteroskedasticity (GARCH) models from econometrics as its building blocks as I attempt to combine the interpretability of econometrics models with the modelling capabilities of neural networks.

- Chapter 6 builds upon the finding from chapter 5 that deep learning can improve forecasting performance of existing volatility models that suffer from the curse of dimensionality. This chapter introduces a data-driven model which generalises to higher dimensions to accommodate larger investment portfolios. Comparison against an existing state-of-the-art deep volatility model is made to highlight the superiority of the proposed architecture.

- Chapter 7 concludes the thesis and outlines future areas of research for other researchers.

1.4 Thesis structure

The thesis flowchart outlines the relationship between the four experiments from inception and proof of concept to final application in a financial setting.


Stochastic Recurrent Neural Network for Multistep Time Series Forecasting:
I introduce the variational autoencoder recurrent neural network (VRNN) time series model
I evaluates the benefit of and feasibility of deep latent time series models
I investigate the performance of a proposed deep latent time series model on multistep time series forecasting
Deep Recurrent Modelling of Granger Causality of Latent Confounding:
I establish from the first experiment the effectiveness of deep latent time series models for forecasting
I leverage the findings from the previous experiment to investigate Granger causality in the presence of a potential confounder by proposing a new framework for nonlinear Granger causality
Neural Generalised Autoregressive Conditional Heteroskedasticity:
Given the effectiveness of deep latent time series models, I apply them in a financial context to predict market volatility
I propose a VRNN GARCH model to perform one step-ahead volatility forecasting with GARCH coefficients being the latent states inferred by the neural network
Variational Heteroskedastic Volatility Model:
Since Neural GARCH can suffer from curse of dimensionality, I propose a VRNN type model for volatility forecasting in higher dimensions with a set of constraints to ensure the outputs of the neural network are valid covariance matrices

Chapter 2

Background Literature

In this chapter I present relevant background literature which serves as the foundations for the theme of this thesis: time series forecasting with deep learning based approaches, and learning relationships between observable and latent dynamics. The surveying of relevant papers here provides motivation for extending and improving existing works. Papers that are applicable to all four above-mentioned experiments are included in this chapter, whilst others specific to each experiment are discussed in their respective chapters.

2.1 Time series forecasting

Time series modelling has been an integral component within academic research and industry, in disciplines such as finance (Malik, 2005), neuroscience (Wang et al., 2018), and causal inference (Marinazzo et al., 2011). A univariate time series is a collection of historical values of a variable of interest, such as the past prices of a stock traded on an exchange. Analysis of historical values allows one to extract important information through tasks such as anomaly detection (Choi et al., 2021), clustering (Procacci and Aste, 2019), classification (Zheng et al., 2014), regime identification (Mari and Mari, 2022), and forecasting (Yin and Barucca, 2021).

Time series forecasting describes the prediction of future value(s) conditioned on historical observations. Traditional methods of forecasting often rely on parametric models informed by domain expertise (Lim and Zohren, 2020); examples include autoregressive (AR) models (Ullrich, 2021), exponential smoothing (Gardner, 2006), and structurual time series models (Harvey, 1990). The use of traditional methods often requires strict assumptions on various time series properties such as stationarity, trend, seasonality, cyclically, and linearity. This approach hence works well when the user possesses detailed understanding about the series being predicted, for example, the forecast of electricity demand requires knowledge about intraday and seasonal patterns.

Recent advancements in computational power and the availability of big data have resulted in an increasing popularity of so-called ”data-driven” models, where the key concept is to leverage the learning capabilities of a machine learning model to approximate the relationships between variables of interest. In this thesis, I focus on artificial neural networks, which due to their strengths in learning complex and nonlinear representations, has achieved significant performance enhancements in fields such as computer vision (Chai et al., 2021), content generation (Iliadis et al., 2022), and natural language processing (Khurana et al., 2023).

2.1.1 Deep learning for time series

The use of deep neural networks (DNN) for time series forecasting has attracted research interest for several reasons (Lim and Zohren, 2020):

- Complex relationships between variables are learnt implicitly during model training. This alleviates the need for manual feature selection and engineering, and could potentially benefit users with insufficient domain expertise.

- A DNN can be constructed with various components and loss functions, which gives the user much flexibility and allows the incorporation of domain expertise when needed. For example, when forecasting a time series for which the inclusion of a longer history is known to improve forecasting performance, one could experiment with a transformer model (Vaswani et al., 2017) over an LSTM (Hochreiter and Schmidhuber, 1997).

- The availability of opensource backpropagation frameworks such as Tensorflow (Abadi et al., 2015) and Pytorch (Paszke et al., 2019) has enabled users to train DNNs and experiment with various components in a time efficient manner.

Many deep learning models have so far been applied to time series forecasting. Examples include convolutional neural network (van den Oord et al., 2016; Koprinska et al., 2018), recurrent neural network (Hewamalage et al., 2021; Hochreiter and Schmidhuber, 1997; Yunpeng et al., 2017; Salinas et al., 2017), multilayer perceptrons (Tank et al., 2021; Liu et al., 2022), generative adversarial networks (Goodfellow et al., 2014; Festag et al., 2022; Brophy et al., 2023), and most recently transformers (Vaswani et al., 2017; Zhou et al., 2021). Each model has their own merits and limitations and it relies on the user to select the optimal model by considering factors such as ease of training and model complexity. In this thesis, I focus mostly on recurrent neural networks (RNN) as I design models to learn the relationship between observable and latent variables at each time step using the hidden state of the RNN.

Time series forecasting can be done for one-step ahead (Zhong and Enke, 2019; Dong et al., 2013) or multi-step ahead (Ferreira and da Cunha, 2020; Khan and Maity, 2022), as well as deterministically (point forecast) (Shen and Shafiq, 2020; Lachiheb and Gouider, 2018) or probabilistically (Hauser et al., 2017; Rangapuram et al., 2018). For a deterministic one-step ahead prediction, a neural network is used to output the prediction $\hat{y}_{t+1} = f_\theta(y_{1:t}, x_{1:t})$, where y_t is the variable of interest at time t, and x_t represents other covariates to be included. The role of a neural network is to approximate as close as possible the relationship f between y and x by adjusting its parameters θ with an optimiser such as ADAM (Kingma and Ba, 2014). For multi-steps ahead prediction, a neural network outputs a sequence of future predictions $\hat{y}_{t+1:t+n} = f_\theta(y_{1:t}, x_{1:t})$, where n is the number of prediction steps.

Contrary to deterministic forecasting, which outputs a single point forecast for every time step, probabilistic forecasting aims to estimate the distribution of future values condition on past history. For one-step ahead forecasting, this involves estimating the distribution $P(y_{t+1}|y_{1:t}, x_{1:t})$, and for multi-step prediction

$P(y_{t+1:t+n}|y_{1:t}, x_{1:t})$. In this thesis, I conduct experiments on all four types of forecasting.

2.2 Learning latent temporal dynamics with deep learning

Applying deep learning to time series forecasting has achieved improved results, with a large body of literature focusing on how best to predict future values of a time series conditioned on its history and other relevant covariates. However, many important time series in the real world are not directly observable. Market regime for example, is a reflection of current market conditions, and predicting the next state allows one to forecast potential market movements. Since these time series are latent, they are often inferred using observational time series. Figure 2.1 shows a hidden process model typically used to describe the relationship between a latent variable z and observed time series y.

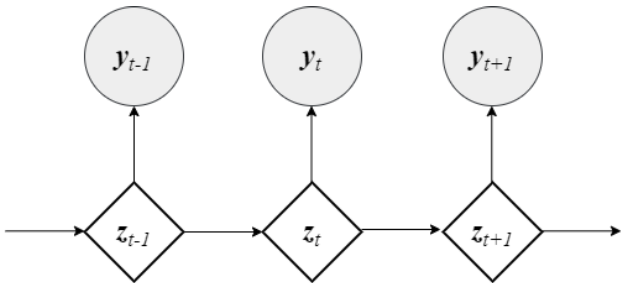


Figure 2.1: A hidden process model with latent variable z and observational variable y.

A hidden process model consists of a transition process describing the temporal evolution of **z**, and an emission/observation process describing the relationship between **z** and **y**. Both transition and emission processes can be deterministic or probabilistic. In the financial domain, the hidden markov model (HMM) has been used extensively to study market regimes (Wang et al., 2020; Nguyen, 2018; Zhang et al., 2019) where $\mathbf{z}_t$ is a multinomial vector of size K (for K market states) with a 1 at the inferred state and 0s elsewhere. However, HMMs are often limited to use cases where z_t is known to be a discrete random variable and when its transition process can be assumed to be markovian, i.e. $P(\mathbf{z}_t|\mathbf{z}_{t:t-1}) = P(\mathbf{z}_t|\mathbf{z}_{t-1})$.

Ultimately, the aim of any inference algorithm is to parameterise the the conditional probability distributions of latent variable given historical values of the observed variable, examples include the filtering distribution $P(\mathbf{z}_t|\mathbf{y}_{1:t})$ and the smoothing distribution $P(\mathbf{z}_t|\mathbf{y}_{1:T})$ (Li et al., 2015).

The classical Kalman filter (Kalman, 1960; Li et al., 2015) is well-known algorithm for handling inference in a state space model when $\mathbf{z}_t$ is a continuous random variable. The transition and emission processes are assumed to be linear and Gaussian. defined by the following equations:

$$\mathbf{z}_t = \mathbf{A}_t\mathbf{z}_{t-1} + \mathbf{B}_t u_t + \varepsilon_t \tag{2.1}$$

$$\mathbf{y}_t = \mathbf{C}_t\mathbf{z}_t + \mathbf{w}_t \tag{2.2}$$

where $\mathbf{y}_t$ is the observed quantity, $\mathbf{A}_t$ is the state transition model, $\mathbf{u}_t$ is a control vector, $\mathbf{B}_t$ is the control model, $\mathbf{C}_t$ is the observation model, and $\varepsilon_t \sim \mathcal{N}(\mathbf{0}, \mathcal{E}_t)$ and $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathcal{W}_t)$ are noise vectors with covariance matrices $\mathcal{E}_t$ and $\mathcal{W}_t$.

In a nonlinear scenario, the above equations become:

$$\mathbf{z}_t = f(\mathbf{z}_{t-1}, \mathbf{u}_t) + \varepsilon_t \tag{2.3}$$

$$\mathbf{y}_t = g(\mathbf{z}_t) + \mathbf{w}_t \tag{2.4}$$

where f and g are known nonlinear functions. Algorithms such as the extended Kalman filter (Li et al., 2015) and particle filters (Godsill, 2019) have been developed to handle non-linear and non-Gaussian transition and emission processes.

There has been an increasing interest to investigate the application of artificial neural networks (ANN) to learn the relationship between latent and observed variables (Rangapuram et al., 2018; Zheng et al., 2017; Krishnan et al., 2015). The motivation stems from the fact that ANNs excel at approximating nonlinear functions and hence provide a natural solution when f and g are complex or unknown. The use of a data-driven method to obtain f and g reduces the need for expertise in the underlying time series. Furthermore, ANNs capable of maintaining a memory of past values, such as an RNN, allows one to bypass the markovian assumption in the transition process and investigate the benefit of including of more lagged values. In fact, the memory of an RNN provides a straightforward way to parameterise the filtering (or smoothing) distribution since it maintains a constant-size hidden state $\mathbf{h}_t$ at every time step. The filtering distribution $P(\mathbf{z}_t|\mathbf{y}_{1:t}) = P(\mathbf{z}_t|\mathbf{h}_t)$ can be obtained by using the RNN hidden state as input, as opposed to the raw values $\mathbf{y}_{1:t}$, which grows linearly in time. More details on the technical workings of RNNs and the inference mechanism will be given in subsequent chapters.

In a static setting, Kingma and Welling (2014) proposed the variational autoencoder (VAE) to learn the relationship between $\mathbf{z}$ and $\mathbf{y}$. The idea is to use neural networks to approximate the emission distribution $P(\mathbf{y}|\mathbf{z})$ and the inference distribution $q(\mathbf{z}|\mathbf{y})$, where $q(\mathbf{z}|\mathbf{y})$ is an approximation of the true posterior $P(\mathbf{z}|\mathbf{y})$. The two neural networks are then trained simultaneously by maximising an evidence lower bound of the log likelihood.

Learning the system described in Eqn 2.3 involves two stages: 1. learning the non-linear functions f and g; 2: inference of $\mathbf{z}_t$ through parameterisation of the posterior distribution $P(\mathbf{z}_t|\mathbf{y}_{1:t})$. There are currently two approaches in existing literature where the transition and observation processes are either coupled or decoupled.

For the coupled approach (Chung et al., 2015; Bayer and Osendorfer, 2014; Fabius and van Amersfoort, 2015; Fraccaro et al., 2016; Karl et al., 2017), a simpler distribution (e.g. Gaussian) is used to approximate the true posterior (variational inference). A VAE is often combined with an RNN to perform sequential inference, with two neural networks parameterising $P(\mathbf{y}_t|\mathbf{z}_{1:t})$ and $q(\mathbf{z}_t|\mathbf{y}_{1:t})$ respectively. The advantage is that all model components can be trained simultaneously, resulting in an end-to-end model for which no human intervention is required between input(history and covariates) and output(predictions). The main drawback is that $q(\mathbf{z}_t|\mathbf{y}_{1:t})$ is only an approximation of the true posterior $P(\mathbf{z}_t|\mathbf{y}_{1:t})$ and hence model performance depends largely on its distribution choice.

For the decoupled approach (Rangapuram et al., 2018; Zheng et al., 2017; Krishnan et al., 2015; Seeger et al., 2017), the true posterior is obtained analytically, or through sampling. An assumption, such as linearity, is placed on the transition dynamics f (or estimated with a neural network). A classical filtering algorithm such as the Kalman filter or particle filter can be used to infer the posterior distribution $P(\mathbf{z}_t|\mathbf{y}_{1:t})$. The main advantage is that the posterior tends to be better approximated by sampling compared to variational inference from the coupled approach. However, this approach is more time consuming since sampling can be computationally demanding, and more expertise is required to choose the optimal filtering algorithm, as well as assumptions on the transition function f.

In this thesis I focus on investigating the coupled approach in the field of time series forecasting, aiming to develop end-to-end and easy to use neural network models to learn the relationships between latent and observable time series. The models serve as powerful tools for forecasting latent variables of interest, e.g. market volatility (experiments 1 and 2), or using learnt relationships between latent and observed variables to better forecast an observable time series (experiments 3 and 4).

Chapter 3

Methodology and algorithms

This section provides an overview of the statistical methods and various neural networks used to make the scientific contributions in this thesis. The three important neural networks of interest are: 1. the multilayer perceptron (MLP) used to learn nonlinear mappings between various features, 2. the recurrent neural network (RNN) used to model temporal dynamics, and 3.the variational autoencoder (VAE) used for the inference and prediction of latent variables.

3.1 Multilayer perceptron

The multilayer perceptron is a feedforward neural network consisting of an input layer, one or many hidden layers and an output layer (Rosenblatt, 1958; ichi Amari, 1993), with several nodes forming each layer of the network. A graphical representation of an MLP with one input layer, one hidden layer and one output layer is shown in Figure 6.2 (figure reference: Bento (2021)).

To illustrate the underlying mechanism of an MLP, we adapt the notations of Ramchoun et al. (2016). Consider an MLP with n_i input features $x_1, x_2, ..., x_{n_i}$, H hidden layers with n_h nodes in each layer, and 1 output feature y. The relationship between the input layer and the first hidden layer is given by

$$h_j{}^1 = f(\sum_{i=1}^{n_i} w_{i,j}{}^0 x_i + b_j{}^1) \tag{3.1}$$

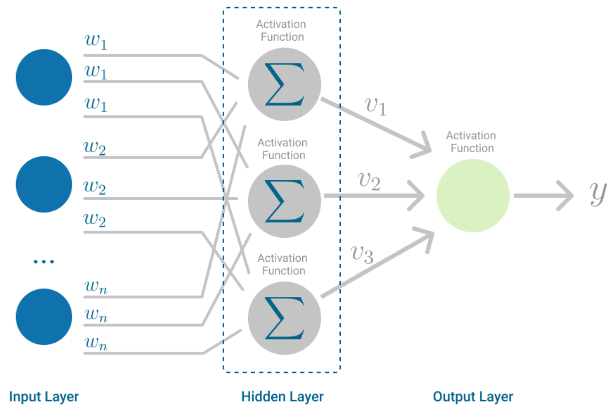


Figure 3.1: A multilayer perceptron with single input, hidden and output layers.

where ${h_j}^1$ is the value at the jth node of the first hidden layer ($j \in [1, 2, ..., n_h]$), ${w_{i,j}}^0$ represents the weight connecting feature x_i to node j in the first hidden layer, and b_j is the bias at node j of the first hidden layer.

A nonlinear activation function f is applied to the sum in Eqn 3.1 before outputting to the subsequent layer. Common choices for f include the rectified linear (ReLU) $f(x) = max(0, x)$, hyperbolic tangent $f(x) = tanh(x)$, sigmoid $f(x) = \frac{1}{1+e^{-x}}$, and many more (Nanni et al., 2022). The choice of the activation function often relies on domain expertise. When the output is a probability for example, the sigmoid function is often used to keep the output between 0 and 1.

The relationship between the hidden layers is described by

$$ {h_j}^l = f(\sum_{i=1}^{n_h} {w_{i,j}}^{l-1} {h_i}^{l-1} + {b_j}^l) \tag{3.2} $$

where ${h_j}^l$ is the value at the jth node of the lth layer and ${h_i}^{l-1}$ represents the value at the ith node from the previous layer $l-1$. Lastly, the relationship between the final hidden layer and the output layer is

$$ y = f(\sum_{i=1}^{n_h} {w_i}^H {h_i}^H + b^o) \tag{3.3} $$

Supervised training of an MLP involves learning the weights w and biases b such that a chosen loss function such as the mean squared error is minimised. A forward pass using data from the training set outputs predicted values from which the loss function can be calculated, the derivatives of the loss function with respect to all parameters are computed through backpropagation, and the parameters are subsequently updated through stochastic gradient descent (Popescu et al., 2009)

$$w_{i,j} = w_{i,j} - \eta \frac{\partial L}{\partial w_{i,j}} \tag{3.4}$$

where η is the step size for parameter updates which is chosen during hyper-parameter tuning, $\frac{\partial L}{\partial w_{i,j}}$ is the partial derivative of the loss function L with respect to the parameter $w_{i,j}$.

3.2 Recurrent neural network

The recurrent neural network is a architecture designed to model sequential data and has achieved tremendous success in areas such as natrual language processing (Khurana et al., 2023).

Similar to multilayer perceptrons, an RNN consists of input, hidden, and output layers. The main difference between an MLP and an RNN is in the way information propagates through the network (Schmidt, 2019). In an MLP, input x_t is mapped to the hidden layer then the output layer. In the context of sequence modelling, this means the MLP does not maintain a memory of past information unless they are passed in at the input layer in the form of lagged values x_{t-1}, x_{t-2} etc. An RNN on the other hand maintains a hidden state h_t at every time steps, which stores past information and is propagated to the next step: $h_{t+1} = f(h_t, x_t)$. The update function f is nonlinear and varies depending on the architecture chosen. In this seciton, the long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (Cho et al., 2014) are presented. RNNs are preferred over MLPs for time series modelling as more historical information can be incorporated without having to increase the dimension of the input at every time step.

The training of an RNN is similar in nature to training an MLP. The weights and biases are updated each time as to minimise a chosen loss function. The underlying algorithm is known as backpropagation through time (Schmidt, 2019), which is an adaptation of the classical backpropagation algorithm to compute the partial derivatives of the loss function with respect to the parameters taking into account the error contributions from each time step. The calculation of partial derivatives of the loss function with the chain rule potentially involves matrix multiplications over long sequences. This gives rise to the vanishing/exploding gradient problem in earlier RNNs, which causes the gradient to approach 0 or diverge and lead to unstable parameter updates.

3.2.1 Long short term memory unit

To overcome the vanishing gradient problem, Hochreiter and Schmidhuber (1997) introduced the long short-term memory units an additive gradient structure and various gates to regulate the flow of information and update the cell state. Figure 5.3 shows a graphical representation of an lstm cell with cell state c_t fro storing information, input i_t output o_t and forget f_t gates to regulate information, and lstm output h_t (figure reference: (Rahuljha, 2020)).

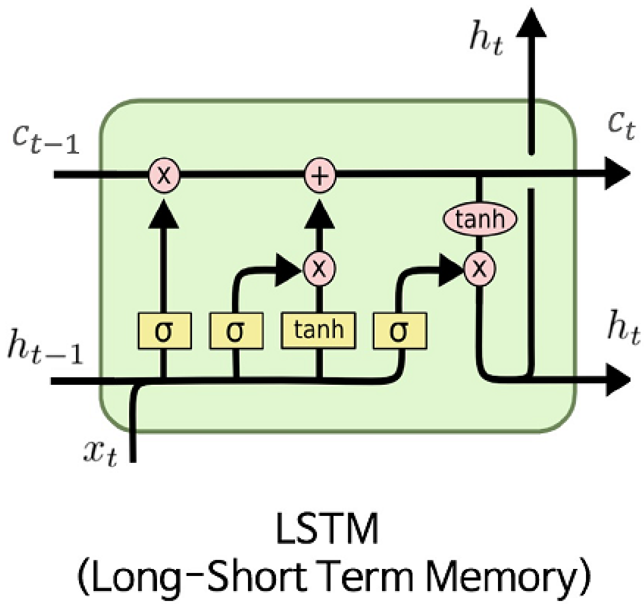


Figure 3.2: An long short-term memory cell.

The update equations for an lstm cell with input x_t, weight matrices W and biases b are given below, where σ represents the sigma function. We adapt the notations in Rahuljha (2020).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.5}$$

$$i_t = \sigma(W_i \cdot [h_{t-1,x_t}] + b_i) \tag{3.6}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_t - 1, x_t] + b_C) \tag{3.7}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.8}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.9}$$

$$h_t = o_t * tanh(C_t) \tag{3.10}$$

3.2.2 Gated recurrent unit

The gated recurrent unit proposed by Cho et al. (2014) is another RNN variant with fewer parameters than the LSTM, consisting of update and reset gates. There has been evidence to show that the GRU offers comparable performance to the LSTM in many sequence modelling tasks (Chung et al., 2014) and hence many studies have opted to use the GRU over the LSTM for faster convergence and reduced overfitting.

A graphical representation of a GRU cell is given in Figure 4.3 (figure reference: Chen (2021)) with input x_t, update gate z_t, reset gate r_t, and cell output h_t.

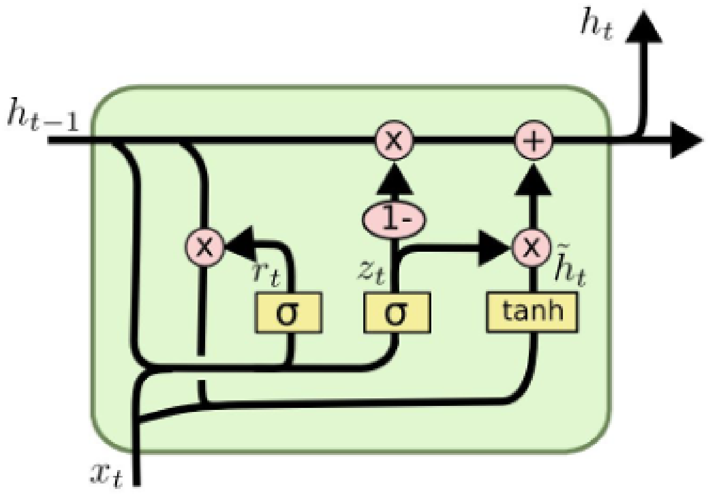


Figure 3.3: A gated recurrent unit cell.

The update equations for a GRU cell is given by

$$z_t = \sigma(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \tag{3.11}$$

$$r_t = \sigma(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \tag{3.12}$$

$$\tilde{h_t} = tanh(W_h \cdot x_t + r_t * U_h \cdot h_{t-1} + b_z) \tag{3.13}$$

$$h_t = z_t * h_{t-1} + (1 - z_t) * \tilde{h_t} \tag{3.14}$$

where W and U are weight matrices, b is the bias vector, and σ is the sigmoid function.

3.3 Variational autoencoder

3.3.1 Variational inference

Consider a latent variable model consisting of a continuous latent random variable z and observed variable x depicted in Figure 4.4 (figure reference: Kingma and Welling (2014)), we are interested in learning the joint distribution $P(x,z) = P(x|z)p(z) = p(z|x)p(x)$. The first part $P(x|z)p(z)$ describes the generative process, given a sample of z from the prior distribution $P(z)$, one can obtain a sample of x from the distribution $P(x|z)$. For the rest of this thesis, $P(x|z)$ is referred to as the decoder.

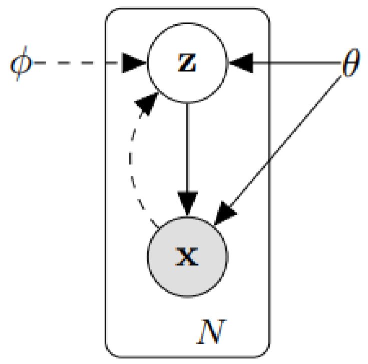


Figure 3.4: Setup of latent variable model.

The distribution $p(z|x)$ is known as the posterior distribution as it describes an estimate of the distribution of the latent variable conditioned on observing the corresponding x value. The posterior distribution is potentially intractable, so one cannot evaluate or differentiate it (Kingma and Welling, 2014). Variational inference is a form of approximate inference where a simpler distribution $q(z|x)$ is used to approximate the true posterior $P(z|x)$.

In this thesis we follow the notations in Kingma and Welling (2014). The generative distribution $P_\theta(x|z)$ has parameters θ, the variational posterior (or approximate posterior) $q_\phi(z|x)$ has parameters ϕ. The aim of variational inference is to seek optimal values of ϕ such that $q_\phi(z|x)$ most closely approximates $P(z|x)$.

To learn the parameters θ of the decoder $P_\theta(x|z)$, the aim is to maximise the marginal likelihood $P_\theta(x) = \int P_\theta(x|z)P_\theta(z)\,dz$, which is intractable when the likelihood function $P_\theta(x|z)$ is complex, e.g. when it is parameterised by a neural network (Kingma and Welling, 2014).

To overcome the intractability of the marginal likelihood $P_\theta(x)$, it can be rewritten as the sum of an evidence lowerbound and the KL divergence (a statistical distance between two distributions) between the approximate posterior $q_\phi(z|x)$ and the true posterior $P_\theta(x|z)$:

$$logP_\theta(x) = L(\theta,\phi) + D_{KL}(q_\phi(z|x)||P_\theta(z|x)) \tag{3.15}$$

where the evidence lowerbound is given by:

$$L(\theta,\phi) = \mathbb{E}_{z\sim q(z|x)}[\log P_\theta(x|z)] - KL(q_\phi(z|x)||P_\theta(z)) \tag{3.16}$$

$L(\theta,\phi)$ is referred to as an lowerbound since $logP_\theta(x) \geq L(\theta,\phi)$ and they are equal if the D_{KL} equals 0, i.e. if the apprixmate posterior exactly equals the true posterior.

Since the marginal likelihood $logP_\theta(x)$ does not depend on ϕ, by maximising the lowerbound $L(\theta,\phi)$ we simultaneously minimise the KL divergence between the apprixmate and true posterior distributions; this means that θ and ϕ are learned jointly during model training.

3.3.2 Amortised variational inference

Kingma and Welling (2014) proposed amortised variational inference, which is the use of neural networks to parameterise the generative distribution $P_\theta(x|z)$ and the approximate posterior $q_\phi(z|x)$.

A graphical representation of a variational autoencoder is given in Figure 4.5 (figure reference: Wikipedia). In this setup, an encoder neural network such as an MLP maps input x to the distribution parameters of the latent space z ($q_\phi(z|x)$), a sample of z is then mapped back to the observation space by a decoder neural network ($P_\theta(x|z)$). The loss function for this setup is the evidence lower bound given in Eqn 3.16.

Since the sampling operation is not differentiable, they introduced the reparameterisation trick to express the sampled z as a differentiable transformation $\tilde{z} = g_\phi(\varepsilon, x)$. For location-scale family of distributions such as Gaussian, this means letting $g = location + scale \cdot \varepsilon$. This differentiable transformation allows θ and ϕ to be learned jointly using stochastic gradient descent.

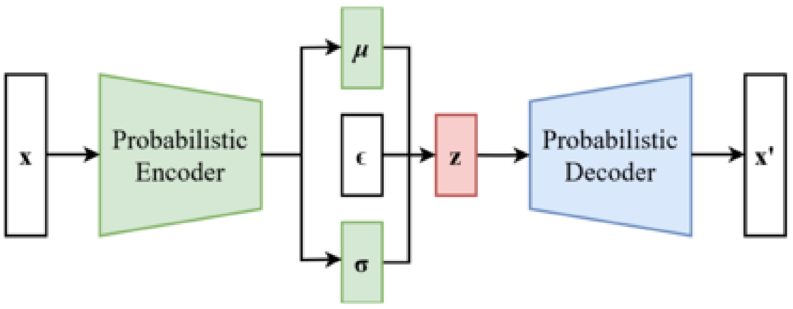


Figure 3.5: Setup of latent variable model.

3.4 Developing deep latent variable models for time series forecasting

In this thesis, a deep latent time series model is developed using the above-mentioned deep learning architectures for various time series modelling tasks.

The model consists of a latent space and an observation space. The mapping to (and from) the latent space to the observation space is done using a variational autoencoder. The transition dynamics of the latent space is modelled with a recurrent neural network. The VAE learns meaningful features about the input time series, and its probabilistic nature facilitates downstream tasks such as generative modelling.

In chapter 4 I develop a VAE-RNN hybrid model suitable for probabilistic time series forecasting with relevant covariates. This section serves as a proof-of-concept chapter to demonstrate the effectiveness of said model on multistep predictions.

In chapter 5, I investigate how a deep latent variable model can be used to learn meaningful relationships between two observed time series. To achieve this, I study the case where the two time series are potentially influenced by a shared confounder. I develop a latent time series model to infer a proxy of the confounder given other relevant time series. Through explicit modelling of the confounder, I study how this can improve the forecasting of one time series given the other.

In chapters 6 and 7 I collect relevant insight from the firs two experiments and design a deep latent variable model to perform conditional volatility forecasting. The first model is constructed by combining a deep latent time series model with traditional GARCH models from the field of econometrics. This model allows the end user with expertise in the matter to pre-select a version of the GARCH model suitable for certain styles effects. The second model is a data-driven model inspired by the proposed architecture in chapter 4, which can be applied to model the returns of higher dimensional portfolios.

Chapter 4

Designing deep latent models for time series forecasting

4.0.1 Introduction and motivation

The aim of this chapter is to explore the VAE-RNN hybrid model in a time series forecasting context. I introduce and test a stochastic variant of the recurrent neural network (RNN) for multistep time series forecasting. The stochasticity is introduced through the combination of a VAE and and an RNN, with linkages between the covariates and the target time series across different time steps. Through a range of experiments I explore the feasibility of the proposed model and its performance against relevant benchmarks.

This chapter is based on the paper Zexuan Yin and Paolo Barucca. Stochastic recurrent neural network for multistep time series forecasting. In *Neural Information Processing*, pages 14–26, Cham, 2021. Springer International Publishing. ISBN 978-3-030-92185-9.

In contrast to many existing forecasting models which learn a relationship between a time series and its lagged values i.e. learning the distribution $P(y_{1:t+1}, x_{1:t}) \propto P(y_{t+1}|y_{1:t}, x_{1:t})$, where x_t are the covariates, I look to extend the latent variable model framework in a deep learning context. This involves performing inference on the latent variable $z_{1:t}$ using observed time series $y_{1:t}$ then modelling the distribution $P(y_{1:t+1}, x_{1:t}, z_{1:t}) \propto P(y_{t+1}|y_{1:t}, x_{1:t}, z_{1:t})$.

Exploring a deep latent variable model for time series brings many benefits and provides an extra dimension for future work in many domains. I will provide two examples here. Firstly, suppose $y_t = r_t$ is the daily log returns of a stock and $z_t = \sigma_t^2$ is the volatility of the stock returns (latent variable). In many cases, it has been established in industry that volatility is easier to predict than the return itself since volatility possesses many well documented stylised facts (such as volatility clustering) (Marra, 2015). In this case, attemping to forecast future returns $r_{t+1} = f(r_{1:t})$ may reveal very little information. However, we could try to analyse the volatility since r_t is often assumed to follow a distribution defined partially by the volatility, for example, in the Gaussian case, one could assume that $r_t \sim \mathscr{N}(0, \sigma_t^2)$. Since volatility is easier to predict than returns, we could obtain valuable information about future returns r_{t+1} by forecasting future volatility instead, using past returns: $\sigma_{t+1}^2 = g(r_{1:t}, \sigma_{1:t}^2)$, where g is a function to be learnt. This prediction parameterises the distribution $P(r_{t+1}|r_{1:t}) = \mathscr{N}(0, \sigma_{t+1}^2)$ under the Gaussian assumption and allows one to gauge one step-ahead potential market movements. We explore volatilty forecasting and deep learning further in Chapter 6.

Another advantage of exploring latent variable time series models is regarding generative modelling. The variational autoencoder introduced in Chapter 3 is an example of a generative model. Since the model learns a distribution over the latent variable z, one can generate more samples of x by first sampling from $P(z)$ and passing it into the decoder $P(x|z)$. The ability to generate samples similar to the original data has many advantages. For example, a bank may want to publish research work on a model trained on client data, however since client data is confidential it would be against the law to release it along with the paper for reproduciblility purposes. A potential solution is to perform the research on AI generated data, which itself can be made open-source along with the paper. The findings in this paper has helped a major investment bank to develop a generative time series model to be used in their electronic trading research.

In this work, I develop a model which uses an RNN (Chung et al., 2014) for temporal modelling and a VAE (Kingma and Welling, 2014) for the inference of associated latent states. I perform forecasting experiments on time series from a wide range of domains to show the effectiveness of the model against chosen benchmarks.

4.0.2 Existing literature on RNNs and VRNNs

Many existing studies have successfully applied recurrent neural networks to model nonlinear temporal dynamics in time series from many domains (Bandara et al., 2019; McNally et al., 2018; Hu et al., 2021). When comparing an RNN to a latent variable time series model (LVM) such as the one introduced in Figure 2.1, the similarities are that both the RNN and LVM are consisted of a transition function and an observation function. The main difference though, is that in an RNN, the transition function (and sometimes the observation function also) is entirely deterministic whereas in an LVM both transition and observation dynamics are stochastic (probabilistic).

Since weights are shared between time steps in an RNN, it applies the same update function to lagged values when forecasting future ones. Chung et al. (2015) argued that this deterministic nature of the RNN means it may not perform as well on time series with high variability. They propose to combine a variational autoencoder (VAE) to an RNN to transform it into a latent variable model. This involves the injection of a latent variable z_t into the update functions of an RNN such that the RNN hidden state h_t is not only a function of historical values of the time series, but also the latent variable. The hidden state h_t after the injection of z_t is now stochastic since z_t is a random variable that is sampled from a learned distribution. This setup combines the benefits of applying an RNN to sequential data and the probabilistic nature of LVMs. The role of the VAE is to perform inference on the injected z_ts given observed data.

The VAE-RNN(VRNN) model consists of a generative model and an inference model. The generative model describes the transition process of the latent time series model and the mapping from the latent space to the observation space. The inference model is responsible for the inference of the latent space conditioned on observed time series.

VRNNs have been shown to perform well on many types such sequential data such as speech, music, and hand writing recognition Chung et al. (2015); Bayer and Osendorfer (2014); Fraccaro et al. (2016); Karl et al. (2017); Fabius and van Amersfoort (2015).

To contribute to existing literature, I explore the VRNN framework specifically on time series forecasting by developing a stochastic gated recurrent unit cell and investigate its performance on multistep forecasting using a wide range of datasets. This will serve as a proof of concept study to see whether the use of the variational autoencoder can learn meaningful relationships between input covariate time series through representation learning. I will also investigate the generative modelling performance of said model through probabilistic forecasting.

4.0.3 Problem formulation and model architecture

For a multivariate dataset comprised of $N+1$ time series, the covariates $x_{1:T+\tau} = \{x_1, x_2, ...x_{T+\tau}\} \in \mathbb{R}^{N\times(T+\tau)}$ and the target variable $y_{1:T} \in \mathbb{R}^{1\times T}$. I refer to the period $\{T+1, T+2, ...T+\tau\}$ as the prediction period, where $\tau \in \mathbb{Z}^+$ is the number of prediction steps and the aim is to model the conditional distribution

$$P(y_{T+1:T+\tau}|y_{1:T}, x_{1:T+\tau}), \tag{4.1}$$

which is achieved through the factorisation of the joint distribution into a product of distributions, and the parameters of the factored distributions are learned using neural networks.

4.0.4 Stochastic GRU cell

Here I introduce the update equations of the stochastic GRU cell, which describes the flow of information through the model:

$$u_t = \sigma(W_u \cdot x_t + C_u \cdot z_t + M_u \cdot h_{t-1} + b_u) \tag{4.2}$$

$$r_t = \sigma(W_r \cdot x_t + C_r \cdot z_t + M_r \cdot h_{t-1} + b_r) \tag{4.3}$$

$$\tilde{h}_t = tanh(W_h \cdot x_t + C_h \cdot z_t + r_t \odot M_h \cdot h_{t-1} + b_h) \tag{4.4}$$

$$h_t = u_t \odot h_{t-1} + (1-u_t) \odot \tilde{h}_t, \tag{4.5}$$

where σ is the sigmoid activation function, z_t is a latent random variable which captures the stochasticity of the temporal process, u_t and r_t represent the update and reset gates, W, C and M are weight matrices, b is the bias matrix, h_t is the GRU hidden state and $\odot$ is the element-wise Hadamard product. This stochastic adaptation can be seen as a generalisation of the regular GRU, i.e. when $C = 0$, it is equivalent to a regular GRU cell (Chung et al., 2014).

4.0.5 The generative model

The role of the generative model is to establish probabilistic relationships between the target variable y_t, the intermediate variables of interest (h_t,z_t), and the input covariates x_t. In this experiment I use neural networks to describe the non-linear transition and emission processes, and I preserve the architectural workings of an RNN - all relevant information at any given time step is encoded in the hidden states that evolve with time. To perform an estimation of the target variable, a mapping is performed to the observation space just like a regular RNN. A graphical representation of the generative model is shown in Fig 4.1. It can be seen that with the inclusion of the random variable z_t, the evolution of the hidden state h_t is no longer deterministic since it depends on the value of z_t which is sampled from a probability distribution. The joint probability distribution of the generative model can be factorised as follows:

$$p_\theta(y_{1:T}, z_{1:T}, h_{1:T} | x_{1:T}) = \prod_{t=1}^{T} p_{\theta_1}(y_t|h_t) p_{\theta_2}(h_t|h_{t-1}, z_t, x_t) p_{\theta_3}(z_t|h_{t-1}) \tag{4.6}$$

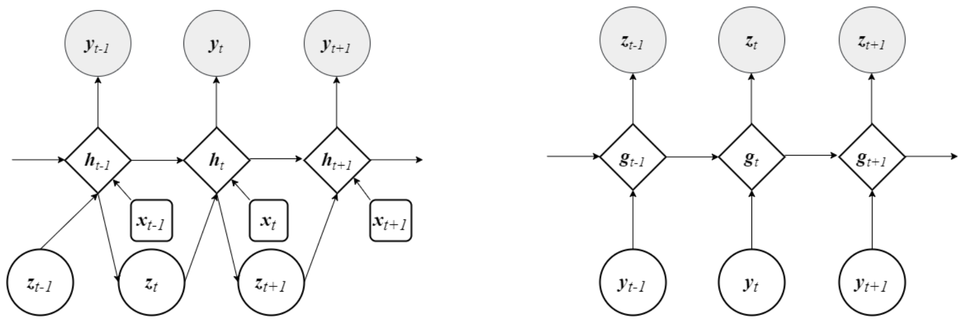
where

$$p_{\theta_3}(z_t|h_{t-1}) = N(\mu(h_{t-1}), \sigma^2(h_{t-1})I) \tag{4.7}$$

$$h_t = GRU(h_{t-1}, z_t, x_t) \tag{4.8}$$

$$y_t \sim p_{\theta_1}(y_t|h_t) = N(\mu(h_t), \sigma^2(h_t)), \tag{4.9}$$

where *GRU* is the stochastic GRU update function given by (4.2)–(4.5). (4.7) defines the prior distribution of z_t, which is assumed to have an isotropic Gaussian prior (covariance matrix is diagonal) parameterised by a multi-layer perceptron. To generate a prediction for y_t, I follow the three equations in order. First, a value of z_t is sampled from $p_{\theta_3}(z_t|h_{t-1})$, which flows through the stochastic GRU cell to obtain h_t. A multilayer perceptron is then used to perform the mapping from h_t to y_t. I refer to the collection of neural network parameters of the generative model as θ, i.e. $\theta = \{\theta_1, \theta_2, \theta_3\}$. I refer to (4.9) as the generative distribution since it describes the process by which the target variable is generated conditioned on the latent variable. This distribution is also parameterised by a multilayer perceptron.



(a) Generative model **(b)** Inference model

Figure 4.1: Proposed generative and inference models

4.0.6 The inference Model

For model traning the aim is to maximise the marginal log-likelihood function $\log p_\theta(y_{1:T}|x_{1:T})$, however the random variable z_t of the non-linear state space model cannot be analytically integrated out. Instead, one needs to maximise the variational lower bound (ELBO) with respect to the generative model parameters θ and inference model parameters ϕ Kingma and Welling (2014). The variational approximation of the true posterior $p(z_{1:T}|y_{1:T})$ can be factorised as follows:

$$q_\phi(z_{1:T}|y_{1:T}) = \prod_{t=1}^{T} q_\phi(z_t|g_t) q_\phi(g_t|g_{t-1}), \tag{4.10}$$

where g_t represents the hidden state of another recurrent neural network used in the inference model, as seen in Fig 4.1. Details on variational inference and approximating posterior distributions has been given in chapter 3.3.

Since the purpose of the inference model is to infer the filtering distribution

$q_\phi(z_t|y_{1:t})$, and that an RNN hidden state contains a representation of current and past inputs, the hidden state g_t is therefore a proxy for the full history $y_{1:t}$. The update equation for g_t and the mapping to the posterior estimate of z_t are given by:

$$g_t = GRU(g_{t-1}, y_t) \tag{4.11}$$

$$z_t \sim q_\phi(z_t|y_{1:t}) = N(\mu(g_t), \sigma^2(g_t)I). \tag{4.12}$$

4.0.7 Training the VRNN time series model

The objective function of the stochastic RNN is the ELBO $L(\theta, \phi)$ given by:

$$\begin{aligned} L(\theta,\phi) &= \int\int q_\phi \log \frac{p_\theta}{q_\phi} dz_{1:T} dh_{1:T} \\ &= \sum_{t=1}^{T} \mathbb{E}_{q_\phi}[\log p_\theta(y_t|h_t)] - KL(q_\phi(z_t|g_t)||p_\theta(z_t|h_{t-1})), \end{aligned} \tag{4.13}$$

where p_θ and q_ϕ are the generative and inference distributions given in (4.6) and (4.10) respectively. It can be seen that this loss function is simply the VAE objective function adapted in a time series context as it is a sum of the losses across all time steps. $\log p_\theta(y_t|h_t)$ represents the output of the generative model and the KL divergence describes the statistical distance between the prior and posterior distributions of z_t.

During training, the posterior network (4.12) is used to infer the latent variable z_t used for reconstructing y_t. During testing, the prior network (4.7) is used to forecast one-step-ahead z_{t+1} before being used to forecast y_{t+1}. The training process seeks to optimise the ELBO with respect to decoder parameters θ and encoder parameters ϕ jointly:

$$(\theta^*, \phi^*) = \underset{\theta,\phi}{\operatorname{argmax}} L(\theta, \phi). \tag{4.14}$$

Since back-propagation is tricky when the model contains a sampling operation, I apply the reparameterisation trick (Kingma and Welling, 2014) to write

$$z = \mu + \sigma \odot \varepsilon, \tag{4.15}$$

where $\varepsilon \sim N(0,I)$ and I sample from ε instead. The KL divergence term in (4.13) can be analytically computed since the prior and posterior of z_t are both assumed to be normally distributed.

4.0.8 Model prediction

Given the last available GRU hidden state h_{last}, prediction window τ and covariates $x_{T+1:T+\tau}$, the model generates predicted target values in an autoregressive manner. Assuming that at every time step the hidden state of the GRU h_t contains all relevant information up to time t. The prediction algorithm of the stochastic GRU is given by Algorithm 1.

```
Input: $\tau, h_{last}, x_{T+1:T+\tau}$
Output: $y_{T+1:T+\tau}$
for $t \leftarrow 1$ to $\tau$ do
    $z_t \sim p_{\theta_3}(z_t | h_{last})$
    $h_t \leftarrow GRU(h_{last}, z_t, x_t)$
    $y_t \sim p_{\theta_1}(y_t | h_t)$
    $h_{last} \leftarrow h_t$
end
```

Algorithm 1: Prediction algorithm for stochastic GRU

4.0.9 Description of experiments

The performance of the proposed model is demonstrated on the following 6 publicly available datasets. For all datasets, checks were done on missing values and variables with more than 20% missing were removed. I performed outlier filtering on datapoints that were beyond 1.5 times the interquartile range. Before model training I performed standardisation on the variables to ensure 0 mean and unit variance.

1. Equity options trading price time series available from the Chicago Board Options Exchange (CBOE) datashop. This dataset describes the minute-level traded prices of an option throughout the day. I study 3 traded options with Microsoft and Amazon stocks as the underlyings where $x_t =$ underlying stock price and $y_t =$ traded option price

2. The Beijing PM2.5 multivariate dataset describes hourly PM2.5 (a type of air pollution) concentrations of the US Embassy in Beijing, and is freely avail-

able from the UCI Machine Learning Repository. The covariates I use are $x_t =$ temperature, pressure, cumulated wind speed, Dew point, cumulated hours of rainfall and cumulated hours of snow, and $y_t =$ PM2.5 concentration. For this experiment I use data from 01/11/2014 onwards

3. The Metro Interstate Traffic Volume dataset describes the hourly interstate 94 Westbound traffic volume for MN DoT ATR station 301, roughly midway between Minneapolis and ST Paul, MN. This dataset is available on the UCI Machine Learning Repository. The covariates I use in this experiment are $x_t =$ temperature, mm of rainfall in the hour, mm of snow in the hour, and percentage of cloud cover, and $y_t =$ hourly traffic volume. I use data from 02/10/2012 9AM onwards

4. The Hungarian Chickenpox dataset describes weekly chickenpox cases (childhood disease) in different Hungarian counties. This dataset is also available on the UCI Machine Learning Repository. For this experiment, $y_t =$ number of chickenpox cases in the Hungarian capital city Budapest, $x_t =$ number of chickenpox cases in Pest, Bacs, Komarom and Heves, which are 4 nearby counties. I use data from 03/01/2005 onwards

To perform probabilistic forecasting with the stochastic recurrent neural network, I generated 500 Monte-Carlo simulations (sampling z_t 500 times) to obtain the predicted confidence intervals, and I took the mean predictions as the point forecasts. I tested the number of simulations from 100 to 1000 and found that above 500, the performance improvements were marginal, and with fewer than 500 it was not possible to obtain realistic confidence intervals for some time series. I compare the model performance against an AR(1) model assuming the prediction is the same as the last observed value ($y_{T+\tau} = y_T$), a standard LSTM model and a standard GRU model. For the performance metric, I normalise the root-mean-squared-error (rmse) to enable comparison between time series:

$$nrmse = \frac{\sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}}{\bar{y}}, \tag{4.16}$$

where $\bar{y} = mean(y)$, $\hat{y}_i$ is the mean predicted value of y_i, and N is the prediction size.

For hyperparameter optimisation, a grid search was performed to select the optimal parameter sets. The sequence length was chosen from $\{10, 20, 30, 40, 50\}$, the latent space size from $\{10, 20, 30, 40, 50, 60\}$, the RNN hidden space size from $\{8, 16, 24, 32, 48, 64, 128, 256\}$, mlp layers from $\{2, 3, 4, 5\}$, and the learning rate from $\{0.1, 0.01, 0.001, 0.0001\}$.

For replication purposes, in Table 4.1 I provide (in order): number of training, validation and conditioning steps, (non-overlapping) sequence lengths used for training, number of prediction steps, dimensions of z_t, h_t and g_t, details about the MLPs corresponding to (4.7)(z_t prior) and (4.12)(z_t post) in the form of (n layers, n hidden units per layer), and lastly the size of the hidden states of the benchmark RNNs (LSTM and GRU). I use the ADAM optimiser with a learning rate of 0.001.

Table 4.1: Model and training parameters

dataset	train	val	cond	seq length	pred	z_t	h_t	g_t	z_t prior	z_t post	RNN hid
Options	300	30	10	10	30	50	64	64	(4,64)	(4,64)	64
PM2.5	1200	200	10	10	30	50	64	64	(4,64)	(4,64)	64
Traffic volume	1000	200	20	20	30	30	128	128	(4,128)	(4,128)	128
Hungarian chickenpox	300	150	10	10	30	50	128	128	(4,128)	(4,128)	128

Table 4.2: nrmse for 30 steps-ahead options price predictions

Option	Description	Ours	AR(1)	LSTM	GRU
MSFT call	strike 190, expiry 17/09/2021	**0.0010**	0.0109	0.0015	0.0015
MSFT put	strike 315, expiry 16/07/2021	**0.0004**	0.0049	0.0006	0.0007
AMZN put	strike 3345, expiry 22/01/2021	**0.0032**	0.0120	0.0038	0.0038

Table 4.3: nrmse for 30 steps-ahead PM2.5 concentration predictions

steps	5	10	15	20	25	30
Ours	**0.1879**	**0.2474**	**0.4238**	**0.4588**	**0.6373**	**0.6523**
AR(1)	0.3092	1.0957	0.7330	0.6846	1.0045	1.1289
LSTM	0.4797	0.6579	0.4728	0.4638	0.8324	0.8318
GRU	0.4846	0.5553	0.4789	0.4919	0.6872	0.6902

Table 4.4: nrmse for 30 steps-ahead traffic volume predictions

steps	5	10	15	20	25	30
Ours	**0.4284**	**0.2444**	**0.2262**	**0.2508**	**0.2867**	**0.2605**
AR(1)	1.2039	1.0541	1.0194	1.0283	1.1179	1.0910
LSTM	0.8649	0.5936	0.4416	0.4362	0.5591	0.5446
GRU	0.8425	0.5872	0.4457	0.4376	0.5510	0.5519

Table 4.5: nrmse for 30 steps-ahead Hungarian chickenpox predictions

steps	5	10	15	20	25	30
Ours	**0.6585**	**0.6213**	**0.5795**	**0.5905**	0.6548	**0.5541**
AR(1)	0.7366	0.7108	0.9126	0.9809	1.0494	1.0315
LSTM	0.7215	0.6687	0.9057	1.0717	0.8471	0.7757
GRU	0.6795	0.6379	0.6825	0.6196	**0.6355**	0.6739

Table 4.6: nrmse of MLP benchmark and our proposed model for 30 steps-ahead forecasts

	MSFT call	MSFT put	AMZN put	PM2.5	Metro	Chickenpox
Ours	**0.0010**	**0.0004**	**0.0032**	**0.6523**	**0.2605**	**0.5541**
MLP	0.0024	0.0005	0.0141	0.7058	0.6059	0.5746

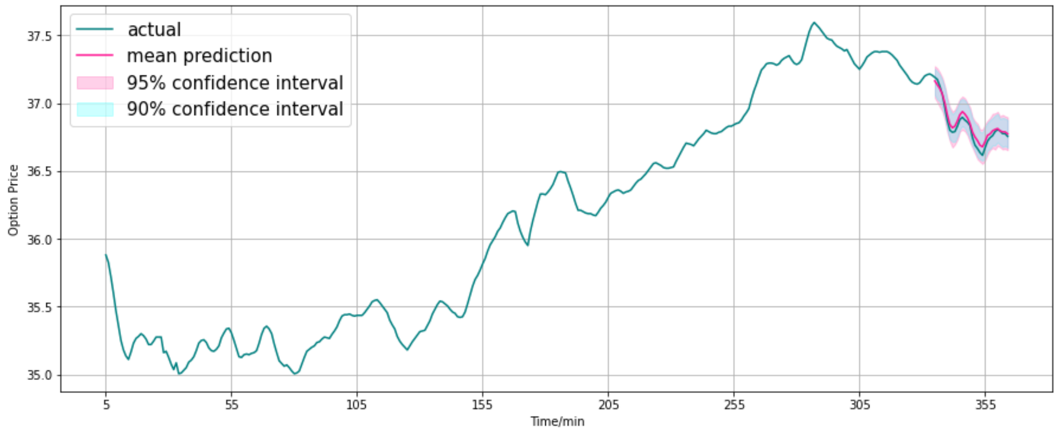


Figure 4.2: MSFT call option, strike 190, expiry 17/09/2021

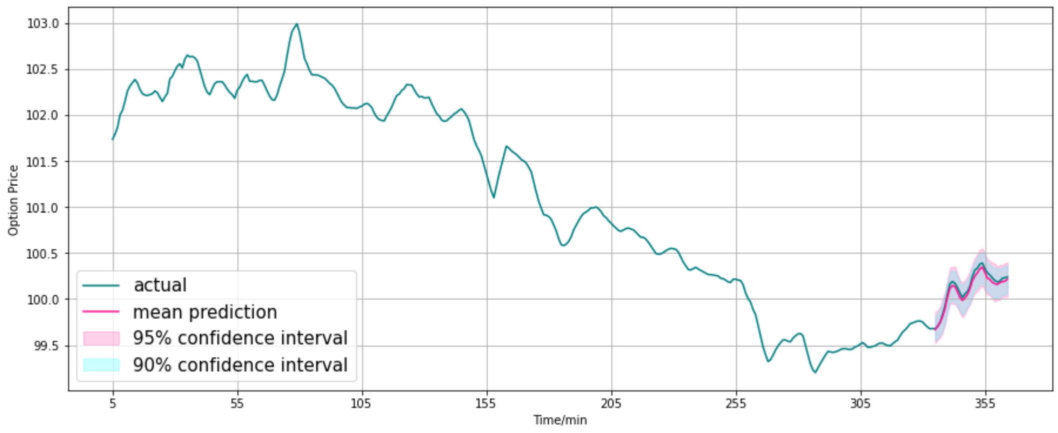


Figure 4.3: MSFT put option, strike 315, expiry 16/07/2021

In Table 4.2–4.5 it can be seen that the nrmse of the stochastic GRU is lower than its deterministic counterpart for all datasets investigated and across all prediction steps. This shows that the proposed method can better capture both long and short-term dynamics of the time series. This approach provides an additional degree of freedom facilitated by the latent random variable which needs to be inferred using the inference network; this allows the stochastic GRU to better capture the stochasticity of the time series at every time step. In Fig 4.6 for example, it can be seen that the model captures well the long-term cyclicity of the traffic volume, and in Fig 4.5 where the time series is much more erratic, the model can still accurately predict the general shape of the time series in the prediction period.

To investigate the effectiveness of the temporal model, I compare the prediction

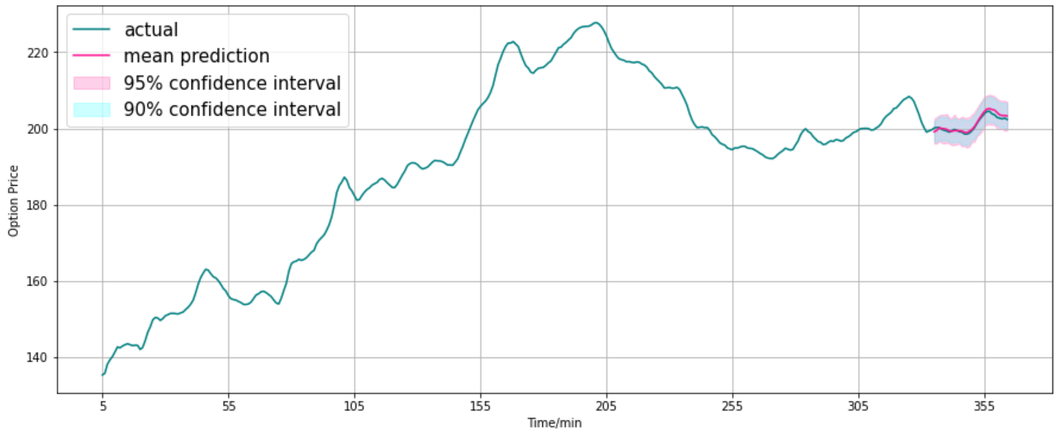


Figure 4.4: AMZN put option, strike 3345, expiry 22/01/2021

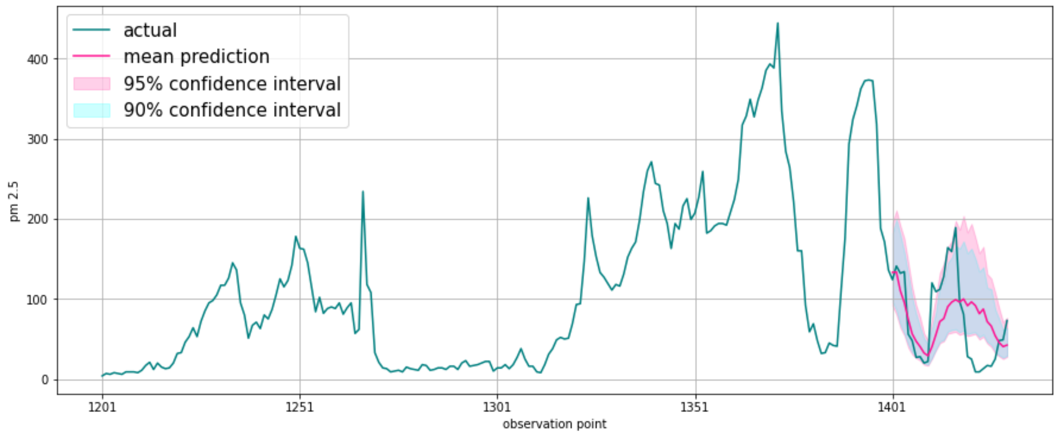


Figure 4.5: PM2.5 concentration forecasts up to 30 steps ahead

errors against a model without a temporal component, which is constructed using a 3-layer MLP with 5 hidden nodes and ReLU activation functions. Since future covariates are being used for prediction (4.1), I aim to verify that the proposed model can outperform a simple regression-type benchmark which approximates a function of the form $y_t = f_\psi(x_t)$; I use the MLP to parameterise the function f_ψ. It can be seen in Table 4.6 that the proposed model outperforms a regression-type benchmark for all the experiments, which shows the effectiveness of the temporal model. It is also worth noting that in the experiments I used the actual values of the future covariates. In a real forecasting setting, the future covariates themselves could be outputs of other mathematical models, or they could be estimated using expert judgement.

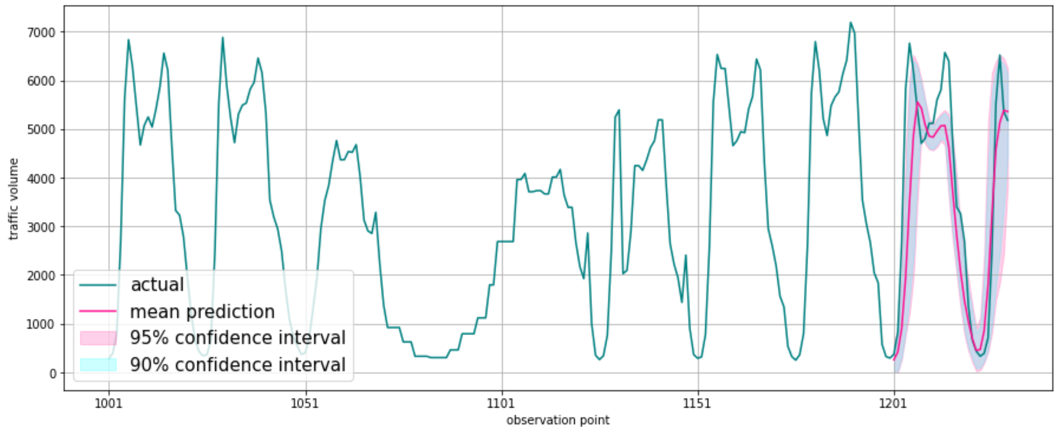


Figure 4.6: Traffic volume forecasts up to 30 steps ahead

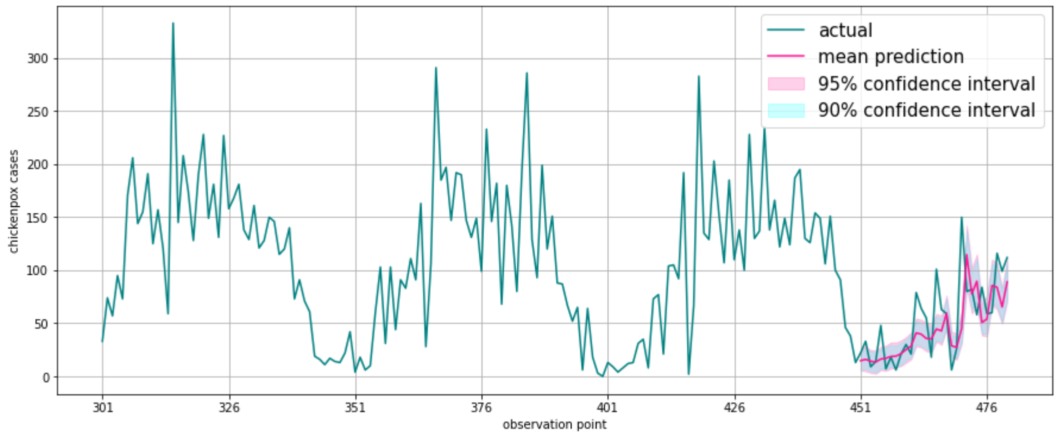


Figure 4.7: Hungarian chickenpox cases forecasts up to 30 steps ahead

4.0.10 Conclusion

In this chapter I presented a stochastic adaptation of the Gated Recurrent Unit which is trained with stochastic gradient variational Bayes. The model design preserves the architectural workings of an RNN, which encapsulates all relevant information into the hidden state, however the adaptation takes inspiration from the stochastic transition functions of state space models by injecting a latent random variable into the update functions of the GRU, which allows the GRU to be more expressive at modelling highly variable transition dynamics compared to a regular RNN with deterministic transition functions. I tested the performance of our model on different publicly available datasets and results demonstrate the effectiveness of the design. Given that GRUs are now popular building blocks for much more complex deep architectures, the stochastic GRU could prove useful as an improved component which can be integrated into sophisticated deep learning models for sequential modelling.

Chapter 5

Investigation of inter-time series relationships learnt by the deep latent variable model

5.1 Introduction and motivation

I have demonstrated in the last chapter that the VAE-RNN setup and the nonlinear latent variable model can be applied successfully in a forecasting setup as it is able to learn complex relationships between the latent and observable spaces. However, the exact relationship the model has learned is unknown since it merely performs a non-linear mapping from one space to another.

One way to check that a deep latent variabel model is indeed capable of learning meaningful relationships in the latent space is to investigate a setting where the latent space is known.

In this chapter, I study the problem of Granger casuality with latent confounding. Since the latent confounder cannot be directly measured, It is of interest to see how well a deep latent variable can approximate the confounder given other observed time series. The advantage of this setup is that the latent space (the confounder) is potentially explainable since they could have practical meaning in the real world.

The content of this chapter is published as Zexuan Yin and Paolo Barucca.

Deep recurrent modelling of granger causality with latent confounding. *Expert Systems with Applications*, 207, 11 2022. ISSN 09574174. doi: 10.1016/j.eswa.2022.118036

Originally proposed in Granger (1969), the Granger causality test is a statistical test used to assess whether a time series contains useful information in predicting future values of another time series (formal definition given in 5.2). This is a popular test adopted by practitioners from fields such as finance (Hiemstra and Jones, 1994) and climate science (Stips et al., 2016) to confirm their hypotheses on the predictability of a time series of interest using another that is readily available to them.

The original formulation of the Granger causality test (Granger, 1969) accounts for linear relationships between time series and the assumption that the cause precedes the effect in time. It is worth noting that X Granger causes Y does not necessarily imply true causality, it merely states that the lagged values of X contains useful information that can be used to predict future values of Y that cannot be found in either lagged values of Y or any other time series.

Traditionally, linear-model based Granger causality has been tested mostly on linear dynamics in the form of a vector autoregressive model (VAR) (Yuan and Shou, 2020), where one regresses the target series against the lagged values of the covariates and assess whether the coefficients are statistically different from zero, and if so, one concludes the absence of a predictive relationship.

Since real world temporal dynamics are rarely linear, several adaptations to model nonlinear Granger causality have been made using for example polynomial autoregression models (Bezruchko et al., 2008) and kernel-based methods (Marinazzo et al., 2011). Model-free approaches such as transfer entropy (Vicente et al., 2011) are able to detect nonlinear dependencies between time series, however they suffer from high variance and require large amounts of data for reliable estimation (Tank et al., 2021). In this work, I follow a recent trend that uses neural networks to infer complex nonlinear predictive dependencies in time series data (Rahimi et al., 2020; Khanna and Tan, 2020; Nauta et al., 2019; Tank et al., 2021; Bussmann et al., 2020; Trifunov et al., 2019; Brouwer et al., 2020; Marcinkevičs and Vogt, 2021; Moraffah et al., 2021).

An important consideration for causal inference from observational time series is confounding bias. A confounder variable affects both cause and effect and therefore must be accounted for to avoid spurious conclusions. Granger causality relies on the causal sufficiency (no latent confounding) assumption (Spirtes and Zhang, 2016) and is known to be biased in the presence of confounding (Peters et al., 2017). Consider the case where the confounder Z affects the cause variable X with lag 2 and the effect variable Y with lag 4, assuming causal sufficiency would lead to the biased conclusion that X Granger causes Y. When all confounders are identified and observed, the multivariate conditional Granger causality tests can be applied (Chen et al., 2006), which relies on the fact that all variables that could have influenced the target time series have been considered in the analysis (Marinazzo et al., 2011).

In reality however, it is rarely possible to identify, let alone measure all the confounders. Nevertheless, one may have access to noisy measurements of proxies for the confounders (Louizos et al., 2017; Pearl, 2010). With access to these "proxies", could they be used as substitutes for the confounders in the analysis, or is it possible to learn something about the underlying confoudners using these proxies are the main research questions of this chapter.

Since the majority of existing works on neural network-based approaches to Granger causality assume causal sufficiency, how best to account for latent confounders is still an open question. In this work, I apply neural networks to infer representations of the latent confounder from the available proxies, which can be used in the subsequent Granger causality tests. The setup I adopt is given in Fig5.1, which is consistent with Louizos et al. (2017). My aim is to establish whether Granger causality exists between X and Y (both influenced by Z) conditioned on the proxies U.

I contribute to nonlinear Granger causality identification by using multilayer perceptrons to infer representations (or substitutes) of the confounder Z from available proxy variables U. With the learned representations and the covariate X as

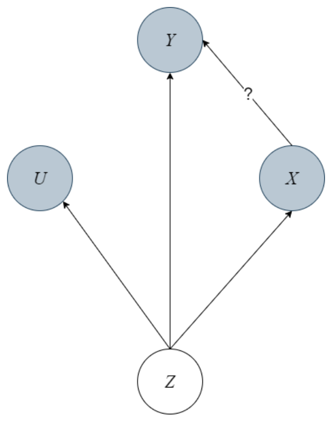


Figure 5.1: Causal graph showing the relationship between effect variable Y, cause variable X, latent confounder Z and proxy variable U.

inputs, I apply a recurrent neural network (RNN) to predict Y. By analysing the performance of this predictive task, one can assess whether the inclusion of X improves the prediction of Y given that the potential effects of the confounder has been accounted for.

5.2 Existing literature on linear and nonlinear Granger causality

The original definition of Granger causality (Granger, 1969) involves linear dynamics studied using a VAR model. For a collection of k time series $X \in \mathbb{R}^{k \times T}$ and $X_t \in \mathbb{R}^k$ a VAR model is defined:

$$X_t = \sum_{l=1}^{L} A^{(l)} X_{t-l} + \varepsilon_t, \tag{5.1}$$

where L is the maximum lag considered, $A^{(l)}$ is a $k \times k$ matrix of coefficients and ε_t is a noise term with zero mean. In the linear regime, time series j does not Granger-cause series i if for all l $A^{(l)}_{ij} = 0$. Tank et al. (2021) generalise the definition of Granger causality for nonlinear autoregressive models:

$$X_{ti} = g_i(X_{1:t,1}, \ldots, X_{1:t,k}) + \varepsilon_{ti}, \tag{5.2}$$

where $X_{1:t,i} = (...,X_{(t-2)i},X_{(t-1)i})$ denotes the history of time series i, and g_i is a nonlinear function mapping the lagged values of other k time series to series i. Granger non-causality is concluded between time series i and j if substituting $X_{1:t,j}$ with another series $X^{'}_{1:t,j}$ ($X^{'}_{1:t,j} \neq X_{1:t,j}$) does not affect the prediction of X_{ti}: $g_i(X_{1:t,1},...X_{1:t,j},...,X_{tk}) = g_i(X_{1:t,1},...X^{'}_{1:t,j},...,X_{tk})$, implying that g_i does not depend on the values of $X_{1:t,j}$.

In Tank et al. (2021) the function g_i is parameterised by a multilayer perceptron (MLP) regularised by group lasso penalties and trained with proximal gradient descent to shrink the input weights of lagged values of non-causal time series to zero. Bussmann et al. (2020) propose a neural additive VAR model with each time series expressed as a sum of nonlinear functions of the other time series. The nonlinear functions are parameterised by MLPs and the additive structure allows the contribution of each time series to be analysed separately.

Nauta et al. (2019) propose an attention based convolutional neural network. The attention mechanism learns which time series are attended to during prediction, and interventions on potential causal time series are performed in the validation phase. Khanna and Tan (2020) infer Granger causal relations from a structured sparse estimate of internal parameters of statistical recurrent units (Oliva et al., 2017) trained for time series prediction.

A popular class of methods involves training two neural network time series prediction models and comparing their performances. One model would accept the past values of the target and exogenous variables as inputs, and the other accepts only the past target values. A statistically significant reduction in prediction error is a sign of Granger causality since this implies that the covariates are adding useful information into the prediction. In existing literature, these prediction models are often different variants of RNNs (Wang et al., 2018; Duggento et al., 2019; Abbasvandi and Nasrabadi, 2019) or MLPs (Orjuela-Canon et al., 2020). The proposed approach in this study falls within this class of methods however training two separate neural networks is inefficient and spurious conclusions could be reached due to differences in neural network hyperparameters. For this reason, I propose to train a single model with two decoders representing $P(Y_t|X_{1:t-1}, Y_{1:t-1})$ and $P(Y_t|X_{1:t-1})$ respectively. These decoders are two separate neural networks trained simultaneously along with the rest of the model.

With the exception of Nauta et al. (2019), all above-mentioned literature assumes causal sufficiency. How best to account for an unobserved confounder in Granger causal analysis is an open question. In Nauta et al. (2019), the model can only detect a latent confounder if it affects cause and effect with equal time lags. In this paper I consider scenarios involving different lags in the causal mechanisms (as given in the example above). I follow a popular approach involving the use of neural networks to infer representations of the latent confounder (a substitute confounder). Louizos et al. (2017) propose a variational autoencoder to recover the joint distribution of the observed and latent variables which they use to estimate the average treatment effect (ATE) in a static setting. Trifunov et al. (2019) adapt the architecture in Louizos et al. (2017) to a time series setting for the estimation of ATE. Bica et al. (2020) propose a recurrent neural network architecture to build a factor model and estimate ATE using the inferred substitute confounders.

Outside of the deep learning domain, different methods can accommodate hidden confounders to different extents. Chu and Glymour (2008) propose additive nonlinear time series model (ANLTSM) which can only deal with hidden confounders that are linear and instantaneous (a lag of 0). Conditional independence based approaches LPCMCI (Gerhardus and Runge, 2020) and SVARFCI(Malinsky and Spirtes, 2018) detect hidden confounders by inferring a special edge type in the partial ancestral graph.

To contribute to existing Granger causality literature, where the approaches mostly assume causal sufficiency, I study the benefit of explicitly modelling the latent confounder through other (related) measurable time series. I demonstrate that information extracted from related time series can serve as a proxy for the confounder which can then be used as an input to better forecast the variable of interest than simply assuming causal sufficiency.

5.3 Designing a deep latent variable for Granger causality

Consider the causal graph in Fig. 5.1 involving a exogenous variable $X \in \mathbb{R}^{1\times T}$, a target variable $Y \in \mathbb{R}^{1\times T}$, a latent confounder $Z \in \mathbb{R}^{1\times T}$ and proxies of the confounder $U \in \mathbb{R}^{n\times T}$, where T is the length of the time series and n is the number of proxies available. The aim is to infer the Granger causal relationship between the confounded pair X and Y.

The proposed approach is a maximum likelihood based latent variable model to learn useful information about the confounder Z from available proxies U, and to model the relationship between Z, X and Y. In practice, the proxy variables are often chosen using expert judgement. Consider a situation where the latent confounder is the socio-economic status of a patient, one could use the zip code or job type of the patient as proxy variables (Louizos et al., 2017).

The learned representation of Z does not contain extra information that is not already found in the time series set U. It is simply extracting relevant information from a potentially large and noisy time series dataset. When there are many proxies to choose from, the mapping from U to Z performs representation learning/dimensionality reduction to learn relevant factors about these proxy variables.

More formally, I follow the assumption in Louizos et al. (2017) that the joint distribution $P(Y,X,Z,U)$ can be approximately recovered from the observations (Y,X,U), which could turn out to be impossible if the confounder has no relation to the observed variables. In this work I adopt the following assumptions: 1. proxy variables are available in abundance to allow recovery of the joint distribution, 2. expert judgement is in place to select appropriate proxies, and 3. (Y,X,U) are potentially complex but learnable functions of Z which we approximate with neural networks. This scenario is termed a ”surrogate-rich setting” in Louizos et al. (2017).

Consider the following nonlinear autoregressive (NAR) model for time series i regressed on the histories of k other time series:

$$X_{ti} = g_i(X_{1:t,1}, \ldots, X_{1:t,k}) + \varepsilon_{ti}, \tag{5.3}$$

with nonlinear function g_i and white noise error term $\varepsilon_{ti} \sim \mathcal{N}(0, \sigma_t^2)$. Since relationships between real world time series are often nonlinear, the definition of nonlinear Granger causality presented by Tank et al. (2021) is adopted in this study.

More formally, time series j does not Granger cause series i if for all $(X_{1:t,1}, ..., X_{1:t,k})$ and all $X'_{1:t,j} \neq X_{1:t,j}$, $g_i(X_{1:t,1}, ...X_{1:t,j}, ..., X_{tk}) = g_i(X_{1:t,1}, ...X'_{1:t,j}, ..., X_{tk})$. This implies that the prediction model g_i does not depend on the history of j ($X_{1:t,j}$), since substituting it with a different time series ($X'_{1:t,j}$) does not affect the prediction of X_{ti}. On the other hand, if series j does Granger cause series i, and j' does not, then the model with $X_{1:t,j}$ as input would lead to a lower prediction error of X_{ti} than using $X'_{1:t,j}$: $(X_{ti} - g_i(X_{1:t,1}, ...X_{1:t,j}, ..., X_{tk}))^2 < (X_{ti} - g_i(X_{1:t,1}, ...X'_{1:t,j}, ..., X_{tk}))^2$.

The nonlinear function g_i can be modelled using a recurrent neural network as it can capture long range dependencies and complex temporal dynamics. The main challenge however is that the definition of Tank et al. (2021) assumes causal sufficiency (no confounding): all k time series are observed. In the presence of confounding, one observes only a subset of k: $(X_{1:t,1}, X_{1:t,2}...) \subseteq (X_{1:t,1}, ..., X_{1:t,k})$; the use of traditional Granger causality tests in this case is known to be biased, as mentioned previously (Peters et al., 2017).

With access to proxy variables U, one can obtain representations of the latent confouder by approximating a function such that $\hat{Z} = f(U) \approx Z$; since f is likely to be a nonlinear function, neural networks could be used for this task. Note that $\hat{Z}$ and U do not need to have the same dimensions, since two proxies could result from the same confounder. Instead, the dimension of $\hat{Z}$ is a hyperparameter that is tuned during model selection.

It is worth mentioning that since Granger causality only accounts for direct causal links (Eichler, 2013), one cannot simply use the proxies in a Granger causality test in place of the latent confounder (Louizos et al., 2017) since it is seen in Fig. 6.1 that there is no direct edge linking U and Y. The sole purpose of U is to learn the properties of Z which can help to better assess the relationship between X and Y. Therefore, one must work backwards along the link $U \rightarrow Z$ to find a substitute confounder $\hat{Z}$ that can be used in place of Z.

Probabilistically, the output of the nonlinear autoregressive model given by (5.3) can be written as:

$$X_{ti} \sim \mathcal{N}(g_i(X_{1:t,1},...,X_{1:t,k}),\sigma_t^2) = P(X_{ti}|X_{1:t,1},...,X_{1:t,k}), \tag{5.4}$$

which is referred to as the predictive distribution of series i at time t conditioned on the histories of itself and other available time series. In this paper, neural networks are used to output the mean (g_i) and variance (σ_t^2).

The proposed approach makes use of multiple recurrent neural networks to parameterise predictive distributions. I introduce two types of prediction models: 1. the full model (with exogenous variables) and 2. the restricted model (without exogenous variables). The full model predictive distribution is defined as $P(Y_{t+1}|Y_{1:t},X_{1:t},Z_{1:t})$. The restricted model distribution is defined as $P(Y_{t+1}|Y_{1:t},Z_{1:t})$. Parameterising the two predictive distributions enables comparison of the predictive performances of two time series prediction models and a statistically significant reduction in prediction error from the restricted model to the full model is a sign of Granger causality $X \rightarrow Y$, or more formally: $(Y_{t+1} - g(Y_{1:t},X_{1:t},Z_{1:t}))^2 < (Y_{t+1} - g(Y_{1:t},Z_{1:t}))^2$.

To parameterise the full-model and restricted-model distributions, recurrent neural networks are used. The proposed model uses gated recurrent units (GRU) (Cho et al., 2014). The architecture of the restricted model is given in Fig. 5.2. Each GRU is characterised by a sequence of hidden states $h_t^{(i)}$ which contains information of time series i up to time t. These hidden states are used as inputs to the multilayer perceptrons (MLPs), which output the distribution parameters of the predictive and inference distributions.

To learn representations of the latent confounder Z using the available proxies U I parameterise the filtering distribution:

$$q_\phi(Z_t|U_{1:t}) = q_\phi(Z_t|h_t^{(U)}). \tag{5.5}$$

The inferred representation $\hat{Z}_t$ of Z_t follows an isotropic Gaussian distribution:

$$\hat{Z}_t \sim q_\phi(Z_t|h_t^{(U)}) = \mathcal{N}(\mu(h_t^{(U)}), \sigma^2(h_t^{(U)})I), \tag{5.6}$$

where the covariance matrix is diagonal. The dimension of $\hat{Z}_t$ is a tunable hyperparameter. The parameters of the filtering distribution are given by

$$(\mu, \sigma) = f_1(h_t^{(U)}), \tag{5.7}$$

where f_1 is a mapping function approximated by an MLP. Let the hidden state $h_t^{(U)} \in \mathbb{R}^{N_{h_U}}$ and $Z_t \in \mathbb{R}^{N_Z}$, the MLP takes as input a vector of size N_{h_U} and outputs a vector of size $N_Z \times 2$ (mean and variance). To ensure positivity of the standard deviation a softplus activation function is applied on the MLP output.

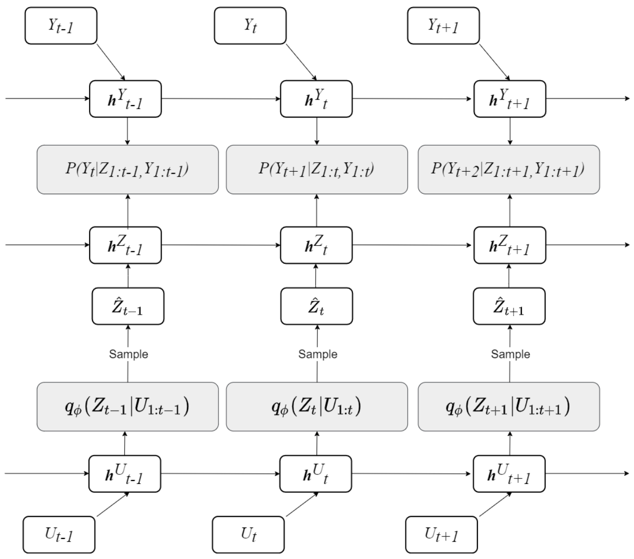


Figure 5.2: Proposed architecture for the restricted model parameterised by multiple recurrent neural networks. q_ϕ is the inference distribution of Z conditioned on the proxy time series, from which samples of the substitute confounder $\hat{Z}$ can be obtained and used to parameterise the predictive distribution of Y.

To avoid the need to train two separate time series prediction models, I propose a dual-decoder setup. The restricted-model distribution is Gaussian and is given as

$$P(Y_{t+1}|Y_{1:t}, Z_{1:t}) = f_2(h_t^{(Y)}, h_t^{(Z)}). \tag{5.8}$$

The full-model distribution is also normal and expressed as

$$P(Y_{t+1}|Y_{1:t},X_{1:t},Z_{1:t}) = f_3(\hat{Y}^{res}_{t+1}, h^{(X)}_t), \tag{5.9}$$

where $\hat{Y}^{res}_{t+1} \sim P(Y_{t+1}|Y_{1:t},Z_{1:t})$ is the predicted value of Y_t from the restricted model and f_2 and f_3 are two MLP models which output the means and variances of the predictive distributions. The proposed dual-decoder setup is shown in Fig. 5.3 and $\hat{Y}^{full}_{t+1} \sim P(Y_{t+1}|Y_{1:t},X_{1:t},Z_{1:t})$. A combination of Fig.5.2 and Fig.5.3 represents the full architecture of the model where the output of the restricted-model serves as one of the inputs to the full-model.

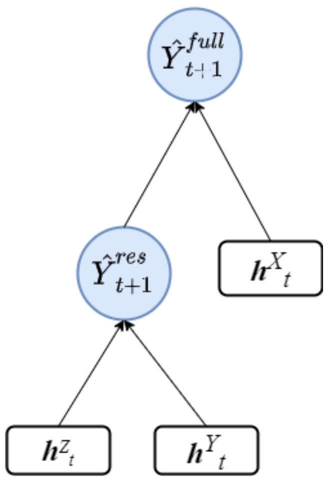


Figure 5.3: Proposed dual-decoder setup where $\hat{Y}^{res}_{t+1}$ is a prediction sample drawn from the restricted-model distribution $P(Y_{t+1}|Y_{1:t},Z_{1:t})$ shown in Figure 6.2.

For model optimisation the following objective function is maximised:

$$L = \sum_{t=1}^{T} \mathbb{E}_{\hat{Z}\sim q_\phi}[\log P_{\theta_1}(Y_t|Y_{1:t-1},X_{1:t-1},Z_{1:t-1}) + \log P_{\theta_2}(Y_t|Y_{1:t-1},Z_{1:t-1})], \tag{5.10}$$

where the first and second terms correspond to the full and restricted model likelihoods respectively, and θ_1 and θ_2 are neural network parameters to be optimised.

To infer the Granger causal relationship between X and Y in the presence of a latent confounder, I assess whether the inclusion of X in the full-model results in a statistically significant reduction in prediction error compared to the restricted model. With access to substitute confounders $\hat{Z}_{1:t}$, a two-sample t-test is performed to establish whether $Y_{t+1} \perp\!\!\!\perp X_{1:t}|\hat{Z}_{1:t}, Y_{1:t}$ (where $\perp\!\!\!\perp$ denotes independence). The mean-squared-error $\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$ was chosen as the error metric.

5.4 Inferring confounded Granger causality in synthetic and real-world datasets

To demonstrate the performance of the proposed approach, I applied it firstly on two arbitrary synthetic datasets with known data generating processes. The nonlinear functions and noise levels have been set arbitrarily. The data generating processes for the two datasets are given by (5.11) and (5.12) respectively. In total, 1000 samples were generated, of which 800 were used for training, 100 for validation and 100 for testing. The dataset was scaled to have 0 mean and unit variance before model training.

5.4.1 Dataset 1

$$\begin{aligned}
Z_t &= tanh(Z_{t-1}) + N(0, 0.01^2)\\
U_t &= Z_t^2 + N(0, 0.05^2)\\
X_t &= \sigma(Z_{t-2}) + N(0, 0.01^2)\\
No\ Granger: Y_t &= \sigma(Z_{t-4}) + N(0, 0.01^2)\\
Granger: Y_t &= \sigma(Z_{t-4}) + \sigma(X_{t-2}) + N(0, 0.01^2)
\end{aligned} \tag{5.11}$$

where the hyperbolic tangent $tanh$ and sigmoid σ functions are used to introduce non-linearity into the system. The noise term is Gaussian of the form $N(\mu, std^2)$.

5.4.2 Dataset 2

The data generating processes for Z, U and X remain the same as in (5.11). The target series Y was generated using:

$$\begin{aligned}
No\ Granger: Y_t &= Z_{t-3}Z_{t-4} + N(0, 0.5^2)\\
Granger: Y_t &= Z_{t-3}Z_{t-4} + X_{t-1}X_{t-2} + N(0, 0.5^2)
\end{aligned} \tag{5.12}$$

5.4.3 River discharge dataset

To investigate the model performance on real-world time series, I use the river discharge dataset provided in Gerhardus and Runge (2020). This dataset describes the average daily discharges of rivers in the upper Danube basin. I consider measurements from the Iller at Kempten as X, the Danube at Dillingen as Y, and the Isar at Lenggries as the proxy variable. All three variables are potentially confounded by rainfall or other weather conditions (Gerhardus and Runge, 2020). The Iller discharges into the Danube within a day, implying an instantaneous causal link $X \to Y$. For the scope of Granger causality considered in this paper, the cause is required to precede the effect in time (Eichler, 2012) so I do not take into account instantaneous causal relationships. I therefore expect no Granger-causal relationship between X and Y. The dataset contains roughly 1000 entries, of which 80% were used for training, 10% for validation and 10% for testing.

Before model training, checks were done to ensure the missing value percentage was less than 20%, and outliers beyond 1.5 times the interquartile range were removed. The data was then scaled to have 0 mean and unit variance before training the neural networks.

5.4.4 Neural network parameters

For hyperparameter optimisation, a grid search approach was adopted. The RNN hidden state dimension was chosen from $\{2,3,4,5,8,16,24,32\}$, the MLP hidden layer and units were chosen from $\{1,2,3,5\}$, the dropout rate was chosen from $\{0.1,0.2,0.3,0.4\}$, and the learning rate was chosen from $\{0.1,0.01,0.001,0.0001\}$.

The GRU hidden states $h_t^{(X)}$,$h_t^{(Y)}$ and $h_t^{(Z)}$ have a dimension of 5, $\hat{Z}_t$ has a dimension of 1 for the synthetic datasets and 2 for the river discharge dataset, the MLPs f_1, f_2 and f_3 given in (5.7,5.8,5.9) respectively contain 1 hidden layer with 5 units for the synthetic datasets and 10 units for river discharge, a dropout rate of 0.3 and ReLU activation functions are chosen. The ADAM optimiser is used with a learning rate of 0.001. The neural networks were trained for 50 epochs. The sequence length used for model training is 20 with a batch size of 10.

5.4.5 Statistical testing

By comparing the sample prediction errors of the full and restricted models, one is able to infer whether a Granger causal relationship between X and Y exists. A two-sample t-test could be used as an additional verification step. For dataset 1&2 (Granger) consider the following null and alternative hypothesis:

$$\begin{aligned} H_0 &: \varepsilon_{full} = \varepsilon_{restricted} \\ H_1 &: \varepsilon_{full} < \varepsilon_{restricted}, \end{aligned} \tag{5.13}$$

where ε_{full} and $\varepsilon_{restricted}$ are the mean prediction errors generated by the full and restricted models respectively. For dataset 1&2 (no Granger) consider the following alternative hypothesis:

$$H_1 : \varepsilon_{full} > \varepsilon_{restricted}. \tag{5.14}$$

The alternative hypothesis is chosen by comparing the sample mean prediction errors computed by the full and restricted models, I.e. the alternative hypothesis in (5.13) is chosen if the mean sample error of the full model is less than that of the restricted model, and vice versa. In cases where the mean sample errors of the two models differ significantly from one another, the statistical test is perhaps redundant. To perform the two-sample t-test, I generate $n = 50$ prediction samples from the restricted and full models and choose a significance level of $\alpha = 0.05$.

In Table 6.1 I provide the prediction errors of the full and restricted models, the p-values of the two-sample t-tests and the Granger causal relationship between X and Y inferred by the proposed model, as well as those inferred by LPCMCI (Gerhardus and Runge, 2020) with $\alpha = 0.05$, maximum lag $L = 5$ and 4 preliminary iterations, and SVAR-FCI (Malinsky and Spirtes, 2018) with $\alpha = 0.05$ and $L = 5$. These are graphical conditional independence based methods for inferring potential causal relationships and are capable of handling latent confounders; thus I have chosen these two models as benchmarks.

It is evident from Table 6.1 that the $p\ value < 0.05$ for all the statistical tests. For dataset 1&2 (no Granger) and the river discharge dataset, I reject the null hy-

Table 5.1: Table showing the prediction errors of the full and restricted models, p-values of two-sample t-tests and the inferred Granger causal relationship given by the proposed model, LPCMCI and SVAR-FCI. The symbol $\times$ denotes that the model finds a Granger non-causal relationship between X and Y.

Dataset	Restricted-model error	Full-model error	p-value	Ours	LPCMCI	SVAR-FCI
dataset 1 (Granger)	$4.99 \times 10^{-2} \pm 6.00 \times 10^{-4}$	$1.76 \times 10^{-2} \pm 5.71 \times 10^{-5}$	< 0.001	✓	✓	✓
dataset 1 (no Granger)	$2.03 \times 10^{-2} \pm 4.00 \times 10^{-4}$	$3.54 \pm 2.00 \times 10^{-4}$	< 0.001	$\times$	✓	✓
dataset 2 (Granger)	$2.07 \times 10^{-1} \pm 7.00 \times 10^{-4}$	$2.03 \times 10^{-1} \pm 9.75 \times 10^{-5}$	< 0.001	✓	$\times$	$\times$
dataset 2 (no Granger)	$1.56 \times 10^{-1} \pm 1.85 \times 10^{-4}$	$1.60 \times 10^{-1} \pm 5.73 \times 10^{-6}$	< 0.001	$\times$	$\times$	$\times$
river discharge	$4.85 \times 10^{-2} \pm 1.50 \times 10^{-3}$	$6.10 \times 10^{-2} \pm 1.12 \times 10^{-3}$	< 0.001	$\times$	$\times$	$\times$

pothesis that the mean prediction errors of the restricted and full models are equal and conclude that the inclusion of X to predict future values of Y results in a higher prediction error and therefore X does not Granger-cause Y. For dataset 1&2 (Granger) I reject the null hypothesis and conclude that the inclusion of X reduces the prediction errors of Y and therefore X Granger-causes Y. The proposed model correctly identifies the correct Granger-causal relationship in all scenarios, whereas LPCMCI and SVAR-FCI identify spurious relationships for dataset 1 (no Granger) and dataset 2 (Granger).

Real-world time series can be highly nonlinear and have different noise levels. The above analysis shows the proposed model is able to identify the Granger-causal relationship for various nonlinear functions and arbitrary noise levels. I investigate the robustness of the proposed approach by varying the signal-to-noise ratio defined as:

$$\gamma = \frac{\frac{1}{T}\sum_{t=1}^{T} |s_t|}{\sigma}, \tag{5.15}$$

where $|s_t|$ denotes the magnitude of the signal (Y_t without the noise term) at t and σ is the standard deviation of the noise term in the data generating process. For dataset 1&2 (Granger) I find the critical γ below which the noise term becomes dominant and the model fails to identify the Granger-causal link between X and Y; to do this I vary the standard deviation σ of the noise term in (5.11) and (5.12). Starting with a rough range of $\gamma = 10$ to $\gamma = 100$, a bisection search strategy to find the critical value γ^*. A *p value* < 0.05 denotes Granger causality inferred by the proposed model. Results are shown in Table 5.2. For dataset 1 (Granger) it can be seen that the critical value γ^* is approximately 59.22 (highlighted in bold), i.e. the

Granger-causal link between X and Y for this set of stochastic time series can only be identified if $\gamma \geq 59.22$; for dataset 2(Granger) $\gamma^* \approx 27.58$.

Lastly, I test the sensitivity of the model output to the sequence length $\tau \in \{4, 6, 8, 10, 12, 14, 16\}$ used in training for dataset 1&2 (Granger). It is seen that all $p\ value < 0.001$, which suggests that the proposed model is able to consistently identify the Granger-causal link given short and long τ used in training. This is desirable as it indicates that model results are not very sensitive to the choice of hyperparameters.

Table 5.2: Sensitivity analysis of model performance with varying signal-to-noise ratio γ.

Dataset 1 (Granger) γ	*p value*	Dataset 2 (Granger) γ	*p value*
10.00	1.00	10.00	1.00
55.00	9.99×10^{-1}	21.25	9.99×10^{-1}
57.81	9.41×10^{-1}	26.88	5.44×10^{-2}
58.51	4.10×10^{-1}	**27.58**	4.73×10^{-3}
59.22	1.93×10^{-2}	28.28	$< 1.00 \times 10^{-3}$
60.63	$< 1.00 \times 10^{-3}$	29.69	$< 1.00 \times 10^{-3}$
66.25	$< 1.00 \times 10^{-3}$	32.50	$< 1.00 \times 10^{-3}$
77.50	$< 1.00 \times 10^{-3}$	55.00	$< 1.00 \times 10^{-3}$
100	$< 1.00 \times 10^{-3}$	100	$< 1.00 \times 10^{-3}$

5.5 Conclusion

In this chapter I proposed a deep-learning based approach to model nonlinear Granger-causality with in the presence of a latent confounder. This is achieved by first learning a representation of the confounder using available proxies, before using these substitutes in place of the confounder for subsequent analysis.

The proposed approach involves the use of multiple recurrent neural networks to parameterise a restricted-model distribution $P(Y_{t+1}|Y_{1:t}, Z_{1:t})$ and a full-model distribution $P(Y_{t+1}|Y_{1:t}, X_{1:t}, Z_{1:t})$. I then generated prediction samples from the two distributions and compared their prediction errors. A two-sample t-test was used as an additional verification step to establish whether the inclusion of X helped to predict future values of Y.

To enable efficient comparison and model training, I propose a dual-decoder setup, which avoids the need to train two separate models (as presented in many existing literature), and I believe this helps to reduce bias resulting from neural network hyperparameter tuning. I demonstrate the effectiveness of the model on both synthetic and real-world datasets, and recognise that a high enough signal-to-noise ratio is required to correctly identify a Granger-causal link.

Chapter 6

Designing deep latent conditional volatility models- part 1

6.1 Introduction and motivation

In chapter 4 I have established that one is able to learn the latent dynamics governing the observed time series using the VAE-RNN hybrid setup. I have shown experimentally that this learned latent model can help to better predict the target time series. This purely data-driven approach however, is a black-box one, and therefore one cannot interpret what has been learned.

In chapter 5, I also attempt to learn a mapping from the observation space to a latent space, however this time the learned latent variable is a representation of the latent confounder. In this context, with access to time series X, Y, and proxies U (and sufficient domain expertise), one can perhaps attempt to infer what it is that the learned substitute is a representation of. For example, if the observational time series are US tech stock prices, one might argue that the learned representations could be a sentiment of the US technology sector, or any other factor known to affect US tech stocks as a whole.

Given the insights generated from the first two experiments regarding the suitability of deep latent time series models to perform forecasting, as well as the experience I have acquired training these types of models, I will attempt to extend this setup to solve real-world forecasting problems.

For the remaining chapters, I focus on developing the VAE-RNN setup in the financial domain. To accomplish this, I design models to perform market volatility forecasting, drawing inspiration from the proposed model in chapter 4.

Volatility forecasting is an important topic that is of interest to the financial industry. However, with market volatility being a latent variable, one could only forecast it using observational time series such as financial returns. This setup offers a compelling real world application of latent variable models and the findings from the first two experiments.

This chapter contains details about the first of the two models, which is suitable for end users comfortable with generalised autoregressive conditional heteroskedasticity (GARCH) models, who may wish to enhance the performance of their GARCH model of choice with deep learning.

6.2 Existing literature on GARCH models

Modelling conditional heteroskedasticity (time-varying volatility) in financial time series such as energy prices (Chan and Grant, 2016), cryptocurrencies (Chu et al., 2017), and foreign currency exchange rates (Malik, 2005) is of great importance to financial practitioners as it allows better decision making with regards to portfolio selection, asset pricing and risk management. In the univariate setting, popular methods include Autoregressive Conditional Heteroskedastic models (ARCH) (Engle, 1982) and Generalised GARCH (GARCH) models (Bollerslev, 1986). ARCH and GARCH models are regression-based models estimated using maximum likelihood, and are capable of capturing stylised facts about financial time series such as volatility clustering (Bauwens et al., 2006). The ARCH(p) model describes the conditional volatility as a function of p lagged squared residuals, and similarly the GARCH(p,q) model include contributions due to the last q conditional variances. Many variants of the GARCH model have been proposed to better capture properties of financial time series, for example the EGARCH (Nelson, 1991) and GJR-GARCH (Glosten et al., 1993) models were designed to capture the so-called leverage effect, which describes the negative relationship between asset price and volatility.

In a multivariate setting, instead of modelling only time-varying conditional variances, for an n-dimensional system, one estimates the $n \times n$ time-varying variance-covariance matrix. This allows investigation into the interactions between the volatility of different time series and whether there is a transmission of volatility (spillover effect) between markets (Bauwens et al., 2006; Erten et al., 2012). Popular multivariate GARCH models include the VEC model (Bollerslev et al., 1988), the BEKK model (Engle and Kroner, 1995), the GO-GARCH model (Van Der Weide, 2002) and DCC model (Christodoulakis and Satchell, 2002; Tse and Tsui, 2002; Engle, 2002).

In this paper I focus specifically on GARCH(1,1) models in the univariate case and the diagonal BEKK(1,1) model in the multivariate case to model daily financial asset returns. I consider several assets classes such as foreign exchange rates, commodities and stock indices. GARCH(1,1) models work well in general practical settings due to their simplicity and robustness to overfitting (Wu et al., 2013).

In traditional GARCH models, the estimated coefficients are constant which imply a stationary returns process with a constant unconditional mean and variance (Bollerslev, 1986). However, there is evidence in existing literature that relaxing the stationary constraint on the returns time series can often lead to a better performance as it allows the model to better capture time-varying market conditions. In Stărică and Granger (2005) the authors modelled daily S&P 500 returns with locally stationary models and found that most of the dynamics were concentrated in shifts of the unconditional variance, and forecasts based on non-stationary unconditional modelling yielded a better performance than a stationary GARCH(1,1) model. Similarly, the authors in Wu et al. (2013) designed a GARCH(1,1) model with time-varying coefficients that followed a random walk process, and they reported better forecasting performances in the test dataset relative to the GARCH(1,1) model.

To this end, I propose univariate and multivariate GARCH models with time-varying coefficients that are parameterised by a recurrent neural network. The proposed method allows the simplicity and interpretability of GARCH models to be combined with the expressive power of neural networks, and this approach follows a trend in the literature that combines classical time series models with deep learning.

In Rangapuram et al. (2018) for example, the authors proposed to parameterise the coefficients of a linear Gaussian state space model with a recurrent neural network, and the latent states were then inferred using a Kalman filter. This approach is advantageous as the neural network allows modelling of more complex relationships between time steps whilst preserving the structural form of the state space model.

Similarly, by preserving the structural form of the BEKK model, one can obtain covariance matrices that are symmetric and positive definite (Engle and Kroner, 1995) without the need of implementing further constraints. I treat the time-varying GARCH coefficients as latent variables to be inferred, and I construct the model with the VAE-RNN setup (Chung et al., 2015; Bayer and Osendorfer, 2014; Krishnan et al., 2017; Fabius and van Amersfoort, 2015; Fraccaro et al., 2016; Karl et al., 2017) to allow efficient structured inference over a sequence of latent random variables.

To contribute to existing GARCH modelling literature, I construct a neural network GARCH hybrid model. This takes advantage of both the explainability of GARCH models and the advanced modelling capabilities of deep latent variable models described in previous sectors. The GARCH coefficients serve as the latent space of the latent variable model and are learned through variational inference.

6.2.1 Univariate GARCH models

The GARCH(p,q) model Bollerslev (1986) for a returns process r_t is specified in terms of the conditional mean equation:

$$r_t \sim \mathcal{N}(0, \sigma_t^2), \tag{6.1}$$

and the conditional variance equation:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i r_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2. \tag{6.2}$$

Under the GARCH(1,1) model, the returns process r_t is covariance stationary with a constant unconditional mean and variance given by $\mathbb{E}[r_t] = 0$ and $\mathbb{E}[r_t^2] = \frac{\omega}{1-\alpha-\beta}$, where $\omega > 0$, $\alpha \geq 0$ and $\beta \geq 0$ to ensure that $\sigma_t^2 > 0$, and $\alpha + \beta < 1$ to ensure a finite unconditional variance. For parameter estimation assuming normal innovations, the following log-likelihood function is maximised:

$$\mathscr{L} = -\sum_{t=1}^{T} \left(\frac{1}{2}\log(\sigma_t^2) + \frac{r_t^2}{2\sigma_t^2}\right) \tag{6.3}$$

To model the leptokurtic (fat-tailed) behaviour of financial returns, the authors in Bollerslev (1987) considered GARCH models with Student's t innovations with the following log-likelihood function to be maximised:

$$\mathscr{L} = -\sum_{t=1}^{T} \left(\log\Gamma\left(\frac{\nu+1}{2}\right) + \log\Gamma\left(\frac{\nu}{2}\right) + \frac{1}{2}log(\nu-2) + \frac{1}{2}\log(\sigma_t^2) + \frac{(\nu+1)}{2}\log\left(1 + \frac{r_t^2}{(\nu-2)\sigma_t^2}\right)\right), \tag{6.4}$$

where $\nu > 2$ is the degree of freedom and Γ is the gamma function.

6.2.2 BEKK model

The BEKK multivariate GARCH model Engle and Kroner (1995) parameterises an n-dimensional multivariate returns process $r_t \in \mathbb{R}^{n\times T}$:

$$r_t \sim \mathscr{N}(0, \Sigma_t), \tag{6.5}$$

$$\Sigma_t = \Omega^T\Omega + \sum_{i=1}^{p} A_i^T r_{t-i} r_{t-i}^T A_i + \sum_{j=1}^{q} B_j^T \Sigma_{t-j} B_j, \tag{6.6}$$

where Σ_t is the $n \times n$ symmetric and positive-definite covariance matrix, Ω is an upper triangular matrix with $\frac{n(n+1)}{2}$ non-zero entries, A and B are $n \times n$ coefficient matrices. In this experiment I consider only the diagonal-BEKK model where A and B are diagonal matrices.

6.3 Designing a neural GARCH model

In this section I introduce the intuition and various components of Neural GARCH models. I focus specifically on univariate and multivariate GARCH(1,1) models as I would like to keep the GARCH model structure as simple as possible and delegate the modelling of complex relationships between time steps to the underlying neural network which outputs the coefficients of the GARCH models. For the rest of this paper, I use the terms (multivariate)GARCH(1,1) and BEKK(1,1) interchangeably when referring to multivariate systems.

In neural GARCH, the coefficients $\{\omega,\ \alpha,\ \beta\}$ in the univariate case and $\{\Omega,\ A,\ B\}$ in the multivariate case are allowed to vary freely with time. This approach allows the model to capture the time-varying nature of market dynamics (Wu et al., 2013). The GARCH(1,1) and BEKK(1,1) models thus become:

$$\sigma_t^2 = \omega_t + \alpha_t r_{t-1}^2 + \beta_t \sigma_{t-1}^2, \tag{6.7}$$

$$\Sigma_t = \Omega_t^T \Omega_t + A_t^T r_{t-1} r_{t-1}^T A_t + B_t^T \Sigma_{t-1} B_t, \tag{6.8}$$

For notation purposes I define the parameter set $\gamma_t = [\omega_t, \alpha_t, \beta_t]^T$ for GARCH(1,1) and $\gamma_t = [\Omega_t, A_t, B_t]^T$ for BEKK(1,1).

In the proposed framework, γ_t is a multivariate normal latent random variable with a diagonal covariance matrix to be estimated at every time step. For GARCH(1,1) this involves an estimation of a vector of size 3 for a model with normal innovations:

$$\gamma_t = \begin{bmatrix} \omega_t \\ \alpha_t \\ \beta_t \end{bmatrix} \sim \mathcal{N}(\mu_t, \Sigma_{\gamma,t}), \tag{6.9}$$

and the vector $[\sigma_{\omega_t}^2, \sigma_{\alpha_t}^2, \sigma_{\beta_t}^2]^T$ represents the diagonal elements of the covariance matrix $\Sigma_{\gamma,t}$. Here I have written the covariance matrix of the parameter set γ_t as $\Sigma_{\gamma,t}$ in order to distinguish it from the covariance matrix of the asset returns Σ_t. For neural GARCH(1,1) with Student's t innovations, γ_t is augmented with the degree of freedom parameter ν_t such that $\gamma_t = [\omega_t, \alpha_t, \beta_t, \nu_t]^T$.

For the multivariate diagonal BEKK(1,1), I adopt a similar methodology. For a system of n assets, γ_t of a model with normal innovations is a vector of size $2n+\frac{n(n+1)}{2}$ (Engle and Kroner, 1995), and with Student's t innovations γ_t is of size $2n+\frac{n(n+1)}{2}+1$. As an example, for a system of 2 assets ($n=2$), the BEKK model is given by:

$$\Sigma_t = \begin{bmatrix} c_{11,t} & 0 \\ c_{21,t} & c_{22,t} \end{bmatrix} \begin{bmatrix} c_{11,t} & c_{12,t} \\ 0 & c_{22,t} \end{bmatrix} + \begin{bmatrix} a_{11,t} & 0 \\ 0 & a_{22,t} \end{bmatrix} \begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \end{bmatrix} \begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \end{bmatrix}^T \begin{bmatrix} a_{11,t} & 0 \\ 0 & a_{22,t} \end{bmatrix} + \begin{bmatrix} b_{11,t} & 0 \\ 0 & b_{22,t} \end{bmatrix} \begin{bmatrix} \sigma^2_{11,t} & \sigma^2_{12,t} \\ \sigma^2_{21,t} & \sigma^2_{22,t} \end{bmatrix} \begin{bmatrix} b_{11,t} & 0 \\ 0 & b_{22,t} \end{bmatrix}, \quad (6.10)$$

where $a_{ij,t}$ is the i,jth element of the matrix A_t, the parameter set γ_t, which also has a multivariate normal distribution, is given by:

$$\gamma_t = [a_{11,t}, a_{22,t}, b_{11,t}, b_{22,t}, c_{11,t}, c_{12,t}, c_{22,t}]^T \quad (6.11)$$

6.3.1 Generative model

The generative model distribution $P_\theta(r_{1:T}, \Sigma_{1:T}, \gamma_{1:T})$ of a general multivariate neural GARCH is presented in Figure 6.1 and given by (6.12). For the univariate case, one simply replaces Σ_t in (6.12) with σ_t^2.

$$P_\theta(r_{1:T}, \Sigma_{1:T}, \gamma_{1:T}) = P(\gamma_0)P(\Sigma_0)\prod_{t=1}^{T} P_\theta(r_t|\Sigma_t)P_\theta(\Sigma_t|\gamma_t, r_{t-1}, \Sigma_{t-1})P_\theta(\gamma_t|\gamma_{t-1}, r_{1:t-1}). \quad (6.12)$$

The initial priors were set to delta distributions, $P(\Sigma_0)$ was centered on the covariance matrix estimated using the training dataset, and $P(\gamma_0)$ was centered on a vector of 1s. The predictive distribution $P_\theta(\gamma_t|\gamma_{t-1}, r_{1:t-1})$ takes as input the information set $\mathscr{I}_{t-1} = \{\gamma_{t-1}, r_{1:t-1}\}$ and predicts the 1-step-ahead value γ_t. For this parameterisation, I use a recurrent neural network to propagate $r_{1:t-1}$ such that:

$$P_\theta(\gamma_t|\gamma_{t-1}, r_{1:t-1}) = P_\theta(\gamma_t|\gamma_{t-1}, h_{t-1}), \quad (6.13)$$

where h_t is the hidden state of the underlying RNN; in the proposed model I use a gated recurrent unit (GRU) (Cho et al., 2014). I then apply a multilayer perceptron which takes as input $\mathscr{I}_{t-1}$ as maps it to the means and variances of the elements in γ_t. In the 2-dimensional example given in (6.11), the estimation is done using:

$$[\mu_{a_{11,t}}, \dots, \mu_{c_{22,t}}, \sigma^2_{a_{11,t}}, \dots, \sigma^2_{c_{22,t}}]^T = MLP_{pred}(\gamma_{t-1}, h_{t-1}), \tag{6.14}$$

and I apply a sigmoid function on the neural network output to ensure that the estimated variances of the elements in γ_t and the GARCH coefficients are non-negative. I have also tested other ways to ensure non-negativity such as using a softplus function, but found that applying a sigmoid function gave the best performance. For neural GARCH with Student's t innovations, it is required that $\nu > 2$ in order to have a well-defined covariance. Since appyling the sigmoid function ensures the estimated coefficients are non-negative, I estimate $\nu' = \nu - 2$ (instead of ν directly) to ensure $\nu > 2$.

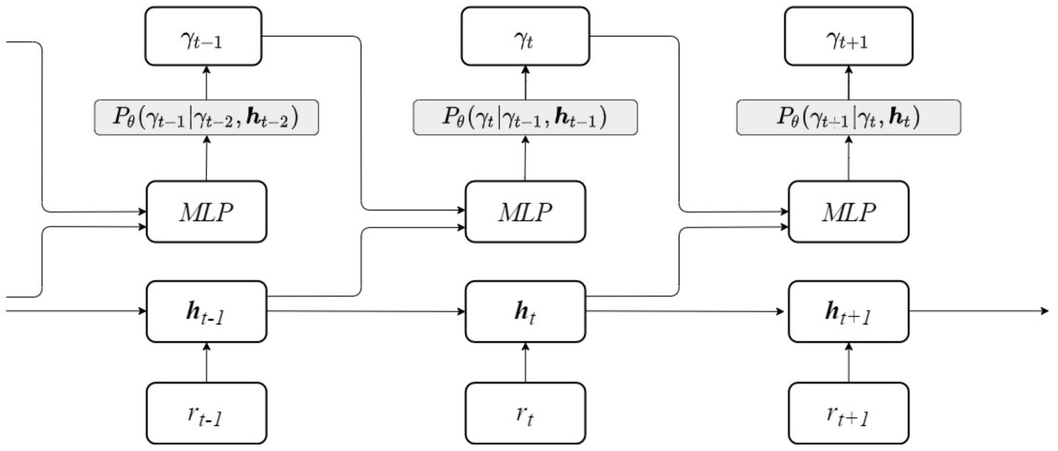
The conditional distribution $P_\theta(\Sigma_t | \gamma_t, r_{t-1}, \Sigma_{t-1})$ is a delta distribution centered on Eqn 6.7 in the univariate case and Eqn 6.8 in the multivariate case as one can calculate the covariance matrix Σ_t deterministically given $\{\gamma_t, r_{t-1}, \Sigma_{t-1}\}$. The distribution $P_\theta(r_t | \Sigma_t)$ is the likelihood function and I have provided the formulas (in the univariate case) in Eqn 6.3 for normal innovations and Eqn 6.4 for Student's t innovations.

6.3.2 Inference model

The inference model distribution $q_\phi(\Sigma_{1:T}, \gamma_{1:T} | r_{1:T})$ is presented in Figure 6.2 and can be factorised as:

$$q_\phi(\Sigma_{1:T}, \gamma_{1:T} | r_{1:T}) = P(\gamma_0) P(\Sigma_0) \prod_{t=1}^{T} q_\phi(\Sigma_t | \gamma_t, r_{t-1}, \Sigma_{t-1}) q_\phi(\gamma_t | \gamma_{t-1}, r_{1:t}), \tag{6.15}$$

where $P(\gamma_0)$ and $P(\Sigma_0)$ are the same as in the generative model, $q_\phi(\Sigma_t | \gamma_t, r_{t-1}, \Sigma_{t-1})$ has the same functional form (a delta distribution) as $P_\theta(\Sigma_t | \gamma_t, r_{t-1}, \Sigma_{t-1})$, however



γ_{t-1}
γ_t
γ_{t+1}
$P_\theta(\gamma_{t-1}|\gamma_{t-2}, \boldsymbol{h}_{t-2})$
$P_\theta(\gamma_t|\gamma_{t-1}, \boldsymbol{h}_{t-1})$
$P_\theta(\gamma_{t+1}|\gamma_t, \boldsymbol{h}_t)$
MLP
MLP
MLP
$\boldsymbol{h}_{t-1}$
$\boldsymbol{h}_t$
$\boldsymbol{h}_{t+1}$
r_{t-1}
r_t
r_{t+1}

Figure 6.1: Generative model of neural GARCH. The generative MLP takes as input $\{\gamma_{t-1}, h_{t-1}\}$ and outputs the estimated means and variances of the elements in γ_t.

γ_t is now drawn from the posterior distribution $q_\phi(\gamma_t|\gamma_{t-1}, r_{1:t})$ where:

$$q_\phi(\gamma_t|\gamma_{t-1}, r_{1:t}) = q_\phi(\gamma_t|\gamma_{t-1}, h_t). \tag{6.16}$$

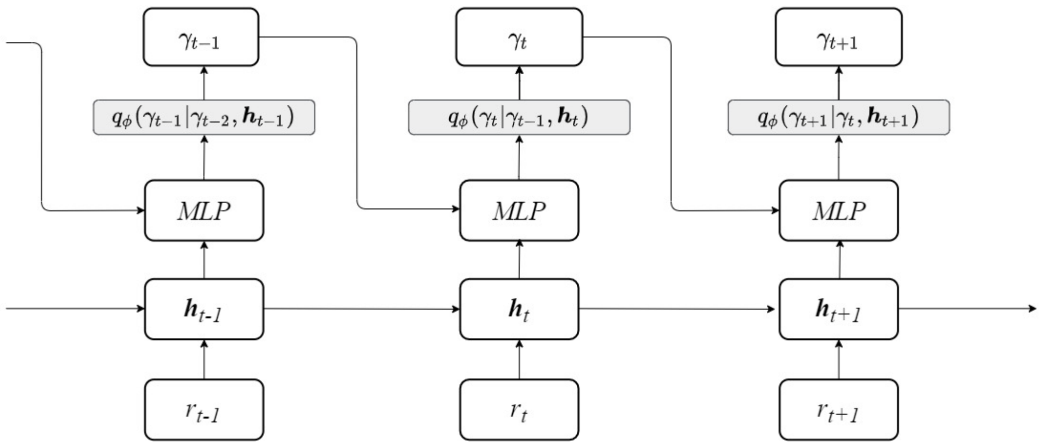
It is worth noting that the generative and inference networks share the same underlying recurrent neural network but uses information at different time steps. The generative model predicts γ_t using the information set $\mathscr{I}_{t-1}$ and the inference model infers γ_t using $\mathscr{I}_t$. The inference MLP (MLP_{inf}) however is different to that of the generative model (MLP_{pred}) and it outputs the posterior estimates of the elements of γ_t:

$$[\mu_{a_{11,t}}, \ldots, \mu_{c_{22,t}}, \sigma^2_{a_{11,t}}, \ldots, \sigma^2_{c_{22,t}}]^T_{post} = MLP_{inf}(\gamma_{t-1}, h_t). \tag{6.17}$$

6.3.3 Optimising a neural GARCH model

For neural network training I optimise the generative and inference model parameters (θ and ϕ) jointly using stochastic gradient variational Bayes (Kingma and Welling, 2014). The objective function is the ELBO defined as:

$$ELBO(\theta, \phi) = \sum_{n=1}^{T} \mathbb{E}_{\gamma_t \sim q_\phi}[_\theta(r_t|\gamma_t)] - KL(q_\phi(\gamma_t|\gamma_{t-1}, r_{1:t})||P_\theta(\gamma_t|\gamma_{t-1}, r_{1:t-1})), \tag{6.18}$$



γ_{t-1}
γ_t
γ_{t+1}
$q_\phi(\gamma_{t-1}|\gamma_{t-2}, \boldsymbol{h}_{t-1})$
$q_\phi(\gamma_t|\gamma_{t-1}, \boldsymbol{h}_t)$
$q_\phi(\gamma_{t+1}|\gamma_t, \boldsymbol{h}_{t+1})$
MLP
MLP
MLP
$\boldsymbol{h}_{t-1}$
$\boldsymbol{h}_t$
$\boldsymbol{h}_{t+1}$
r_{t-1}
r_t
r_{t+1}

Figure 6.2: Inference model of neural GARCH. The inference MLP outputs the posterior estimate of γ_t conditioned on available information up to time t.

and I seek:

$$\{\theta^*, \phi^*\} = \underset{\theta,\phi}{\operatorname{argmax}} ELBO(\theta, \phi). \tag{6.19}$$

6.3.4 Obtaining neural GARCH predictions

Neural GARCH produces 1-step-ahead conditional volatility predictions. Given $\mathscr{I}_t = \{\gamma_t, \Sigma_t, r_{t:t}\}$, I use Eqn 6.14 to obtain the prediction of the parameter set γ_{t+1} given by MLP_{pred}. I then obtain the estimate of Σ_{t+1} deterministically using Eqn 6.8. To estimate Σ_{t+2}, one should now have access to r_{t+1}, and can therefore obtain the posterior estimate of γ_{t+1} using Eqn 6.17 and predict Σ_{t+2} using the posterior estimate of Σ_{t+1}. This posterior update is crucial as it ensures all up-to-date information is used to predict the next covariance matrix.

6.4 Performance of neural GARCH on financial time series

I test neural GARCH on a range of daily asset log returns time series covering univariate and multivariate foreign exchange rates (20 pairs), commodity prices (brent crude, silver and gold) and stock indices (DAX, S&P, NASDAQ, FTSE100, Dow Jones). These datasets have been made available at (Yin, 2021). I provide a brief data description in Table 6.1. Analysis was done to ensure the datasets used contained less than 20% missing values, as well as no outliers beyond 1.5 times interquartile range. Standard scaling was then done to ensure the variables had 0 mean and unit variance.

Table 6.1: Description of asset log returns time series analysed in the experiments.

Dataset	N Time Series	Frequency	Observations	Date Range
Foreign exchange	20	daily	3128	05/08/2011 - 05/08/2021
Brent crude	1	daily	2065	05/08/2013 - 05/08/2021
Silver & gold	2	daily	3109	05/08/2011 - 05/08/2021
Stock indices	5	daily	2054	05/08/2013 - 05/08/2021

For model training, I split each time series such that 80% was used in training, 10% for validation and 10% for testing. The optimal parameters were selected with a grid search approach. The RNN hidden state was chosen from $\{8, 16, 24, 32, 64.128\}$, the MLP hidden layer number from $\{1, 2, 3, 4, 5\}$, and the MLP hidden layer node from $\{8, 16, 32, 64\}$.

The underyling recurrent neural network (GRU) has a hidden state size 64, the generative and inference MLPs (MLP_{pred} and MLP_{inf}) are both 3-layer MLPs with 64 hidden nodes and ReLU activation functions.

For univariate time series, I compare the performance of six models: GARCH(1,1)-Normal, GARCH(1,1)-Student's t, Neural-GARCH(1,1) and Neural-GARCH(1,1)-Student's t, EGARCH(1,1,1)-Normal and EGARCH(1,1,1)-Student's t. Although neural GARCH is an adaptation of the GARCH(1,1) model, I include the EGARCH(1,1,1) model as a benchmark as it is capable of accounting for the asymmetric leverage effect: negative shocks lead to larger volatilities than positive shocks. I would like to investigate whether the data driven approach of neural GARCH allows it to model the leverage effect without the explicit dependence on an asymmetric term as in an EGARCH model. For multivariate time series, I compare the performance of multivariate GARCH(1,1) (BEKK(1,1)) with normal and Student's t innovations against their neural network adaptations. I evaluate the model performance using the log-likelihood of the test dataset.

In Tables 6.2, 6.3, 6.4 and 6.5 I provide the log-likelihoods evaluated on the test dataset for commodity prices, stock indices, and univariate and multivariate foreign exchange time series. I have highlighted the best model for each time series in bold.

For commodity prices, I observe that EGARCH(1,1,1)-Student's t is the best performer on Brent crude, whilst Neural-GARCH(1,1)-Student's t performs best on silver and gold price returns.

For stock indices I observe that Neural-GARCH(1,1)-Student's t performs best on the DAX AND Dow Jones indices whilst EGARCH(1,1,1)-Student's t performs best on S&P500, NASDAQ and FTSE 100. The fact that the neural GARCH models perform better than EGARCH on some datasets shows that the proposed data-driven approach can learn to accommodate many but not all scenarios of the leverage effect, and therefore in cases where EGARCH outperforms, there are benefits associated with the direct modelling of the asymmetric effect.

For univariate foreign exchange time series, I observe that the Neural GARCH variants outperform traditional GARCH models on 16 out of 20 time series, and where neural GARCH outperforms, Neural-GARCH(1,1) with normal innovations performs better on 5/16 time series and Neural-GARCH(1,1)-Student's t performs better on 11/16 time series.

Table 6.2: Test log-likelihoods for commodity price time series. Best result highlighed in bold, higher log-likelihood is better.

Time series	GARCH(1,1)-Normal	GARCH(1,1)-Student's t	Neural-GARCH(1,1)	Neural-GARCH(1,1)-Student's t	EGARCH(1,1,1)-Normal	EGARCH(1,1,1)-Student's t
BRENT	-298.738	-298.689	-307.921	-295.895	-299.966	−292.798
SILVER	-554.595	-551.936	-541.713	−514.476	-572.780	-581.834
GOLD	-462.28	-450.752	-473.074	−421.566	-462.857	-468.509

Table 6.3: Test log-likelihoods for stock index time series.

Time series	GARCH(1,1)-Normal	GARCH(1,1)-Student's t	Neural-GARCH(1,1)	Neural-GARCH(1,1)-Student's t	EGARCH(1,1,1)-Normal	EGARCH(1,1,1)-Student's t
DAX	-261.275	-268.944	-259.321	−244.190	-257.767	-266.163
SNP	-300.849	-298.614	-308.559	-295.934	-300.577	−284.841
NASDAQ	-327.547	-326.401	-331.539	-320.387	-334.237	−312.366
FTSE	-324.437	−314.480	-326.572	-315.606	-322.425	−311.135
DOW	-298.406	-302.196	-315.164	−284.247	-292.974	-293.486

For multivariate foreign exchange time series, I observe that Neural-BEKK(1,1)-Student's t is the best performer on 8/9 time series considered. Across different assets it is seen that the Student's t version of Neural GARCH consistently performs better than the traditional GARCH models as well as Neural GARCH with normal innovations. This suggests that a model with Student's t innovation does indeed model the leptokurtic behaviour of financial time series returns better than a model with normal innovations. This finding is in line with findings from existing literature (for example Bollerslev (1987) and Heracleous (2007)).

Table 6.4: Test log-likelihoods for univariate foreign exchange time series.

Time series	GARCH(1,1)-Normal	GARCH(1,1)-Student's t	Neural-GARCH(1,1)	Neural-GARCH(1,1)-Student's t	EGARCH(1,1,1)-Normal	EGARCH(1,1,1)-Student's t
AUDCAD	−397.251	-402.582	-409.553	-398.645	-397.776	-473.302
AUDCHF	-311.566	-308.029	−293.853	-294.010	-309.295	-312.965
AUDJPY	-346.024	-350.401	-353.213	−335.945	-346.478	-354.095
AUDNZD	-303.986	-318.345	-307.44	−301.514	-303.627	-322.777
AUDUSD	-423.602	-424.594	-432.518	−422.753	-424.498	-425.807
CADJPY	-351.749	-359.545	−349.209	-349.842	-350.460	-362.875
CHFJPY	-238.566	-241.360	-215.536	−208.710	-230.120	-253.050
EURAUD	-338.378	-344.922	-347.995	−336.604	-337.481	-347.259
EURCAD	-347.177	-359.499	−345.989	-347.730	-346.547	-366.701
EURCHF	-277.643	-153.502	-156.567	−142.963	-275.073	-321.051
EURGBP	-366.187	-378.950	-373.515	−364.619	-364.727	-389.416
EURJPY	-266.674	-278.327	-267.374	−256.341	-262.667	-290.897
EURUSD	-332.917	-347.818	−330.471	-334.488	-334.178	-361.348
GBPAUD	−335.530	-346.944	-353.800	-344.842	-335.812	-353.034
GBPJPY	-330.030	-348.729	-337.981	−324.559	-329.013	-359.506
GBPUSD	−418.593	-431.554	-423.460	-419.658	-420.534	-441.162
NZDUSD	−415.648	-416.944	-425.841	-417.380	-416.094	-417.153
USDCAD	-408.008	-416.483	−404.614	-413.507	-406.735	-419.863
USDCHF	-315.963	-303.351	-276.461	−260.177	-282.682	-308.410
USDJPY	-295.295	-304.539	-291.419	−277.477	-294.519	-318.100

Table 6.5: Test log-likelihoods for multivariate foreign exchange time series.

Time series	GARCH(1,1)-Normal	GARCH(1,1)-Student's t	Neural-GARCH(1,1)	Neural-GARCH(1,1)-Student's t
EURGBP,EURCHF	-643.521	-558.275	-523.725	−513.214
GBPJPY GBPUSD	-629.950	-656.198	-649.221	−605.305
AUDCHF AUDJPY	-534.49	-522.934	-497.726	−477.992
EURGBP,EURUSD,EURJPY	-920.085	-959.420	-985.156	−917.907
USDCAD,USDCHF,USDJPY	-1008.821	-998.041	-990.601	−957.912
EURGBP,GBPJPY,USDJPY	−916.957	-943.66	-1011.435	-966.806
GBPAUD,GBPJPY,GBPUSD	-971.522	-991.8238	-1037.296	−967.500
EURCHF,EURGBP,EURJPY,EURUSD	-1196.477	-1127.192	-1105.298	−1078.165
AUDJPY,AUDCHF,EURCHF,GBPJPY	-1505.540	-862.995	-865.471	−783.955

In order to evaluate whether the models' performances across different time series are statistically significant, I plotted a critical difference (cd) diagram by following the approach of the authors in Ismail Fawaz et al. (2019) where a Friedman test at $\alpha = 0.05$ Friedman (1940) was first used to reject the null hypothesis that the four models are equivalent and have equal rankings, and then a post-hoc test was done using a Wilcoxon signed-rank test Wilcoxon (1945) at the 95% confidence level. The critical diagram shows the average rankings of the models across different datasets.

In Figure 6.3 I show the cd plot for univariate time series. A bold horizontal line indicates no significant difference amongst the group of models that are on the line. In the univariate experiments I observe no significant difference amongst the group: EGARCH(1,1,1)-Student's T, GARCH(1,1)-Student's T and Neural-GARCH(1,1); likewise, there is also no significant difference amongst the group: GARCH(1,1)-Student's T, Neural-GARCH(1,1), GARCH(1,1)-Normal and EGARCH(1,1,1)-Normal. I also observe that on average, GARCH(1,1)-Normal and EGARCH(1,1,1)-Normal perform significantly better than EGARCH(1,1,1)-Student's T. I establish that Neural-GARCH(1,1)-Student's t is the best performer overall on the univariate datasets, and it significantly outperforms the other models with an average rank of 1.8929.

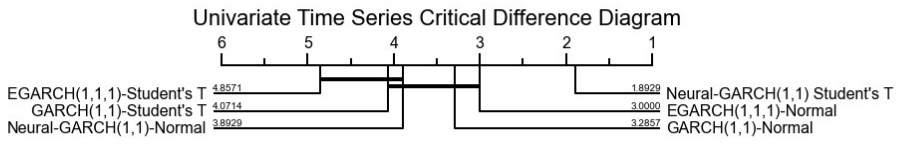


Figure 6.3: Critical difference diagram of the univariate experiments. A horizontal bold line indicates no significant difference amongst the group of models. Neural-GARCH(1,1)-Student's t is the best performer in the univariate experiments.

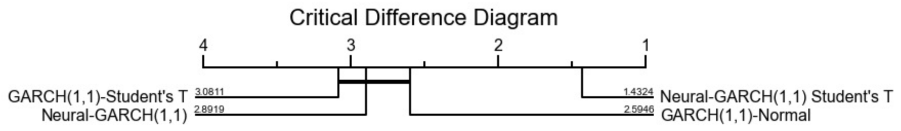


Figure 6.4: Critical difference diagram showing the average rankings of GARCH(1,1) and Neural-GARCH(1,1) with normal and Student's t innovations on all time series experiments. Neural-GARCH(1,1)-Student's t is the best-performing model with an average rank of 1.4324.

In Figure 6.4 I show the cd plot constructed using all the time series experiments (univariate and multivariate). The aim is to compare the class of traditional GARCH(1,1) models against their neural network adaptations. I observe that there is no significant difference between GARCH(1,1)-Student's t, Neural-GARCH(1,1) and GARCH(1,1)-Normal, and it can be established that Neural-GARCH(1,1)-Student's t is the best performer overall with an average ranking of 1.4324.

For a GARCH(1,1) model, the returns process is often assumed to be stationary with a constant unconditional mean and variance. Neural GARCH(1,1) relaxes this stationary assumption. The unconditional variance of Neural-GARCH(1,1) in the univariate case

$$\sigma_t^2 = \omega_t + \alpha_t r_t^2 + \beta_t \sigma_{t-1}^2 \tag{6.20}$$

is obtained by taking the expectation of Eqn 6.20 (since $r_t = \mu_t + \sigma_t \varepsilon$ and assuming

$\mu_t = 0$):

$$\begin{aligned}\mathbb{E}[r_t^2] &= \mathbb{E}[\omega_t + \alpha_t r_{t-1}^2 + \beta_t \sigma_{t-1}^2] \\ &= \omega_t + \alpha_t \mathbb{E}[r_{t-1}^2] + \beta_t \mathbb{E}[\sigma_{t-1}^2] \\ &= \omega_t + (\alpha_t + \beta_t)\mathbb{E}[r_{t-1}^2].\end{aligned} \tag{6.21}$$

For a GARCH(1,1) model with constant coefficients $\{\omega, \alpha, \beta\}$, $\mathbb{E}[r_t^2] = \mathbb{E}[r_{t-1}^2]$ (constant unconditional variance) and therefore $\frac{\omega}{1-\alpha-\beta}$. With Neural-GARCH(1,1), $\mathbb{E}[r_t^2] \neq \mathbb{E}[r_{t-1}^2]$ however one can assume that the parameters $\{\omega_t, \alpha_t, \beta_t\}$ change gradually with no sudden jumps and therefore $\mathbb{E}[r_t^2] \approx \mathbb{E}[r_{t-1}^2]$ (Bri, 2017) and approximate the time-varying unconditional variance of Neural-GARCH(1,1) with $\mathbb{E}[r_t^2] \approx \frac{\omega_t}{1-\alpha_t-\beta_t}$ with $\alpha_t + \beta_t < 1$.

Results from the analysis of the Neural-GARCH(1,1) coefficients show a consistent pattern when compared to GARCH(1,1) models. I provide an example for the currency pair USDCHF in Figure 6.5, which shows the time-varying parameter set $\{\omega_t, \alpha_t, \beta_t\}$ of Neural-GARCH(1,1) against the constant set $\{\omega, \alpha, \beta\}$ of GARCH(1,1). It can be observed across that Neural-GARCH(1,1) consistently estimates a higher value for ω and α, and a lower value for β. In Figure 6.6 I show the zoomed-in plots of the Neural-GARCH(1,1) coefficients shown in Figure 6.5 for the currency pair USDCHF. It can be seen that the coefficients follow well-behaved time-varying behaviour and similar dynamics is observed across all three parameters.

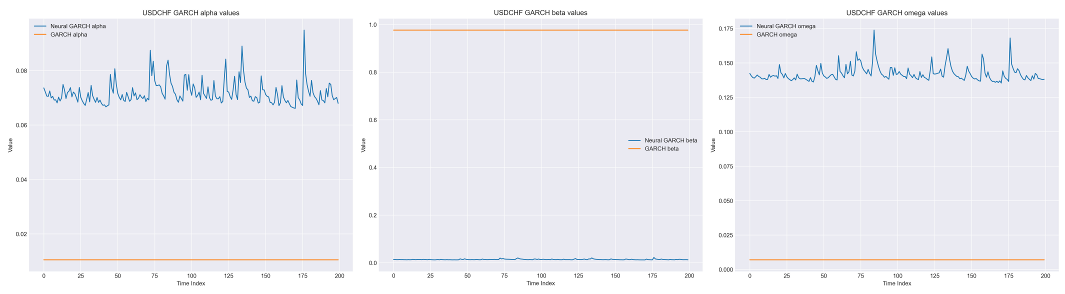


Figure 6.5: Plots of Neural-GARCH(1,1) coefficients against GARCH(1,1) coefficients. The blue line represents the Neural-GARCH(1,1) α_t(left), β_t(middle) and ω_t(right), and the orange line shows the GARCH(1,1) coefficients.

Having time-varying coefficients allows one to model financial returns time

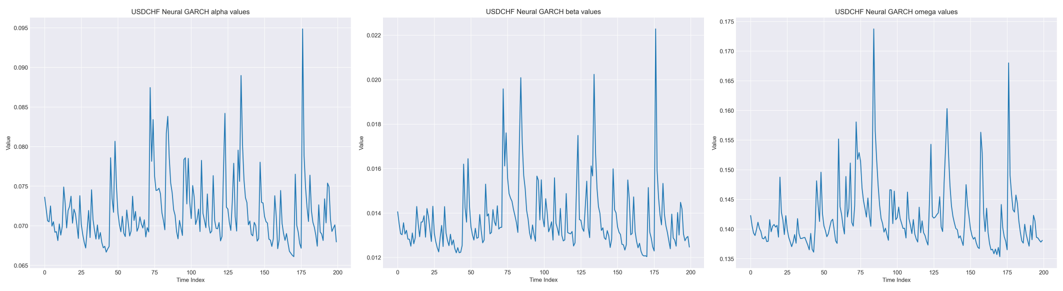


Figure 6.6: Zoomed-in plots of the Neural-GARCH(1,1) coefficients shown in Figure 6.5 for USDCHF.

series as a non-stationary process with a 0 unconditional mean but time-varying unconditional variance. Similarly, the authors in Stărică and Granger (2005) reported that by relaxing the stationarity assumption on daily S&P 500 returns and using locally stationary linear models, a better forecasting performance was achieved, and in their analysis they showed most of the dynamics of the returns time series to be concentrated in shifts of the unconditional variance.

The proposed model provides a data-driven approach to modelling the returns process. During model training no constraints were placed on the neural network parameters, however it can be observed in Figure 6.6 that the model nonetheless outputs time-varying coefficients that satisfy the condition $\alpha_t + \beta_t < 1$, which is required for the model to have a well-defined unconditional variance.

6.5 Conclusion

In this chapter I propose neural GARCH: a neural network adaptation of the univariate GARCH(1,1) and multivariate diagonal BEKK(1,1) models to model conditional heteroskedasticity in financial time series. The approach consists of a recurrent neural network that captures the temporal dynamics of the returns process and a multilayer perceptron to predict the next-step-ahead GARCH coefficients, which are then used to determine the conditional volatilities.

The generative model of neural GARCH makes predictions based on all available information, and the inference model makes updated posterior estimates of the GARCH coefficients when new information becomes available.

This is a successful adaptation of the neural latent variable model in a financial setting. In this setup, the latent space is the coefficients of the chosen GARCH model. Once the coefficients are obtained, the volatility can be calculated deterministically using the chosen GARCH model. This therefore makes the latent space interpretable and allows one to combine domain knowledge and nonlinear modelling to design more powerful volatility models.

However, neural GARCH models are still GARCH models by nature, and therefore suffer from curse of dimensionality. This makes the model less ideal in cases where the portfolio has higher dimensions. In the next experiment, I explore an alternative setup to handle covariance matrix forecasting in higher dimensions.

Chapter 7

Designing deep latent conditional volatility models- part 2

7.1 Introduction and motivation

In chapter 6 I introduced an approach to combine the explainability of traditional GARCH models and deep learning models to forecast conditional volatility. This approach has two limitations: 1. it is suitable for only experience professionals with detailed knowledge of GARCH models; 2. it suffers from the curse of dimensionality and cannot be used for larger portfolios.

In this chapter, I design an alternative model for end users either with limited knowledge on GARCH models and hence favor a data-driven approach, or those who may wish to forecast the conditional volatility of a larger (n>5) portfolio.

The design of the proposed model draws inspiration of that introduced in chapter 4; the covariance matrix of a portfolio is modelled as the latent space of the deep latent time series model.

Financial time series is known to exhibit heteroskedastic behaviour - time-varying conditional volatility (Engle and Patton, 2007; Poon and Granger, 2003). Being able to model and predict this behaviour is of practical importance to professionals in finance for the purpose of risk management (Christoffersen and Diebold, 2000; Long et al., 2020), derivative pricing (J.Duan, 1995), and portfolio optimisation (Ranković et al., 2016; Escobar-Anel et al., 2022).

It is well documented in literature that (conditional) volatility is forecastable on hourly or daily frequencies (Christoffersen and Diebold, 2000). On a univariate level, this involves predicting time-varying variances of asset returns; this predictability can be attributed to the so-called volatility clustering phenomenon: large (small) changes in asset price are often followed by further large(small) changes (Engle and Patton, 2007; Fama, 1965; Schwert, 1989). On a multivariate level (involving a portfolio of assets), volatility forecasting involves estimating conditional covariances between asset pairs in addition to conditional variances of the assets. One could argue that some of this predictability comes from the so-called spillover effect: the transfer of shock between different financial markets (Jebran et al., 2017; Hassan and Malik, 2007; Du et al., 2011). An effective multivariate volatility model therefore needs to capture both intra and inter-time series dynamics.

The traditional way of performing volatility forecasting with GARCH models works well in low dimensions (fewer than 5 assets) due to their simplistic parametric nature. When modelling a portfolio of assets however, it becomes difficult to capture complex dependencies between the various assets that make up the portfolio. Furthermore, the O(n^5) computational complexity of GARCH models prevents them from being easily applied in higher dimensions. To this end, I propose the Variational Heteroskedastic Volatility Model (VHVM), which uses a neural network structure (variational autoencoder + recurrent neural network) to simultaneously model the temporal and inter-asset relationships between the returns of the portfolio instruments. The proposed model performs well relative to commonly used multivariate volatility models, as well as the current state of the art neural network volatility model, with a computational complexity of O(n^2).

7.2 Stochastic volatility models and existing deep learning attempts

Traditional volatility forecasting models can be divided into two main categories: Generalised AutoRegressive Conditional Hertoscedasitcity (GARCH) (Bollerslev, 1986; Nelson, 1991; Glosten et al., 1993) and Stochastic Volatility (SV) (Jacouier et al., 1994; Chan and Grant, 2016) models. The two competing classes of models rely on different underlying assumptions (Luo et al., 2018). GARCH models describe a deterministic relationship between future conditional volatility and past conditional volatility and squared returns. SV models assume that conditional volatility follows a latent autoregressive process. Although there is no general consensus that GARCH is always superior to SV (or vice versa), there is some evidence that SV models are more flexible in modelling the characteristics of asset returns (Chan and Grant, 2016; Shapovalova, 2021). Nonetheless, the popularity of GARCH models seems to surpass that of SV models due to several reasons. Firstly, GARCH models are easier to fit than SV models. The parameters of GARCH are obtained using maximum likelihood estimation, whereas for SV models one needs to obtain samples from an intractible posterior distribution using methods such as Markov chain Monte Carlo (MCMC), which works well when the number of parameters is small, however the convergence can be slow in larger models (Shapovalova, 2021). Secondly, there is an abundance of open-source software (packages) for GARCH models, such as fGarch (Wuertz et al., 2017) and rugarch (Galanos and Kley, 2022) in the programming language R, and arch (Sheppard et al., 2022) in Python. For SV models however, there was no go-to package for model estimation until the release of stochvol and factorstochvol (Hosszejni and Kastner, 2021) in R.

It is worth mentioning that GARCH models too suffer from the curse of dimensionality. For a portfolio of n assets, the computational complexity of GARCH models scales with $O(n^5)$, which makes it impossible fit to beyond a portfolio of roughly 5 assets (Wu et al., 2013).

In recent years, the application of deep learning models for time series forecasting has achieved state of the art performances in many domains (Bandara et al., 2019; Böse et al., 2017; Li et al., 2018). In this experiment I specifically focus on multivariate asset returns time series, a natural research direction would be to investigate whether deep learning models can capture complex dependencies between different assets across time. There are two main obstacles in this task. Firstly, the conditional volatility (covariance matrix) is a latent variable and must be inferred using observational data (Luo et al., 2018). Secondly, for a matrix to be a valid covariance matrix, it must be symmetric and positive definite (Engle and Kroner, 1995). How to impose these constraints on a neural network such that its outputs are valid covariance matrices is a challenging task.

To tackle the first challenge, I adopt a recent trend that combines a variational autoencoder (VAE) and a recurrent neural network (RNN) (a VRNN) to allow efficient structured inference over a sequence of continuous latent random variables (Chung et al., 2015; Bayer and Osendorfer, 2014; Krishnan et al., 2017; Fabius and van Amersfoort, 2015; Fraccaro et al., 2016; Karl et al., 2017). The use of a VAE (and hence variational inference) translates posterior approximation into an optimisation task which can be solved using a neural network trained with stochastic gradient descent. The use of an RNN allows information from previous steps to be used in the modelling and forecasting of the latent variable in future steps. This promising framework has been explored in the neural GARCH experiment as well as in Luo et al. (2018). The authors in Luo et al. (2018) proposed a purely data driven approach to volatility forecasting under the VRNN framework which I used as a baseline model in the results section.

To tackle the second challenge, one possible approach is to combine traditional econometrics models with deep learning models. Traditional econometrics models have well understood statistical properties and neural networks can be used to enhance the predictive power of the model. In neural GARCH for example, I use a neural network to parameterise the time-varying coefficients of the BEKK(1,1) model, which is a multivariate GARCH model proposed by Engle and Kroner (1995). The BEKK model produces symmetric and positive definite covariance matrices by design and therefore no other constraints need to be applied to the neural network output. As mentioned previously however, many econometric models suffer from curse of dimensionality. Instead, I follow the approach in Dorta et al. (2018) and design a neural network such that it outputs the Cholesky decomposition of the precision matrix (inverse of the covariance matrix) at every time step. I will show later on how this guarantees symmetry and positive definiteness. (Engle and

Kroner, 1995)

For a financial asset with price S_t at time t, its log returns are computed as $r_t = log(S_t) - log(S_{t-1})$. The returns process r_t can be assumed to have a conditional mean $\mathbb{E}[r_t|I_{t-1}] = 0$ and a conditional variance $\mathbb{E}[{r_t}^2|I_{t-1}] = {\sigma_t}^2$, in other words $r_t|I_{t-1} \sim \mathcal{N}(0, \sigma_t^2)$. The information set I_t describes all relevant available at time t: $I_t = \{r_{1:t}, {\sigma_{1:t}}^2\}$. The time variation of conditional variance σ is known as heteroscedasticity and the aim of volatility forecasting is to model this behaviour (Engle and Kroner, 1995).

7.2.1 Recap on GARCH models

ARCH (Engle, 1982) and GARCH (Bollerslev, 1986) models have been dominating the field of volatility forecasting since the late 1900s due to their simple model form, explainability, and ease of estimation. Many GARCH model variants have since been proposed to account for well-known stylised facts about financial time series such as volatility clustering and leverage effect (Engle and Patton, 2007). The EGARCH (Nelson, 1991) and GJR-GARCH (Glosten et al., 1993) models for example, were designed to specifically accommodate the leverage effect. The most general GARCH(p,q) model proposed by Bollerslev (1986) describes a deterministic relationship between future conditional volatility, past conditional volatility and squared returns:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i r_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2, \tag{7.1}$$

where p and q are lag orders of the ARCH and GARCH terms, under which the returns process r_t has an unconditional mean $\mathbb{E}[r_t] = 0$ and unconditional variance $\mathbb{E}[r_t^2] = \frac{\omega}{1-\alpha-\beta}$.

In total, there are many hundreds of GARCH model variants, however there exists little consensus on when to use which GARCH models as their performances tend to vary with the nature and behaviour of the time series being modelled, for example, the leverage effect is frequently observed in stock returns but rarely seen in foreign exchange currency returns (Engle and Patton, 2007). In Hansen and Lunde (2005) the authors compared the performances of 330 GARCH variants on Deutsche Mark-US Dollar exchange rates and IBM returns and found that the GARCH(1,1) was not outperformed by any other model in the foreign exchange analysis. In the IBM stock returns analysis however, the authors found that GARCH(1,1) was inferior to models that explicitly accounted for the leverage effect. It has thus become common practice to explore various GARCH variants for the same task (Chan and Grant, 2016; Chu et al., 2017; Malik, 2005).

For a portfolio of assets, models tend to be multivariate generalisations of the univariate GARCH model. In additional to modelling conditional variances for each asset, one also needs to model time varying covariances between different asset pairs. The output for a multivariate GARCH model is a time-varying covariance matrix which describes the instantaneous intra and inter-asset relationships. Notable examples of multivariate GARCH models include the VEC model (Bollerslev et al., 1988), the BEKK model (Engle and Kroner, 1995), the GO-GARCH model (Van Der Weide, 2002), and the DCC-GARCH model (Christodoulakis and Satchell, 2002; Tse and Tsui, 2002; Engle, 2002).

For this experiment, I used the DCC-GARCH (dynamic conditional correlation) model as a multivariate GARCH baseline. The key difference between DCC-GARCH and BEKK (a popular multivariate GARCH) is that BEKK assumes constant conditional correlation between assets, i.e. the change in the covariance between two assets with time is due to the changes in the two variances (but the conditional correlation is constant) (Huang et al., 2010). The constant conditional correlation (CCC) assumption is rather crude since during different market regimes one would expect the correlation between assets to vary. The DCC-GARCH is a generalisation of a CCC-GARCH that accounts for dynamic correlation. During the estimation procedure, various univariate GARCH models are fit for the assets, followed by estimations of the parameters for conditional correlation.

When fitting a multivariate GARCH model under the assumption of normal innovations ($r_t \sim \mathcal{N}(0, \Sigma_t)$), one seeks to maximise the multivariate normal log likelihood function (Bauwens et al., 2006):

$$\mathscr{L}(\theta) = -\frac{1}{2}\sum_{t=1}^{T}(log|\Sigma_t| + r_t^T\Sigma_t^{-1}r_t), \tag{7.2}$$

which becomes computationally expensive in higher dimensions since one is required (for a portfolio of n assets) to invert an $n \times n$ covariance matrix Σ_t for every time step.

One solution to alleviate this burden is to directly work with the precision matrix (inverse covariance matrix) instead: $P = \Sigma^{-1}$. Setting a neural network output to be a precision matrix rather than a covariance matrix allows one to compute the log likelihood straightaway; hence bypassing the expensive matrix inversion step during model training. When the actual covariance matrix is required during the testing phase, one could simply invert the precision matrix to obtain the covariance matrix (Luo et al., 2018; Dorta et al., 2018).

7.2.2 Stochastic volatility as an alternative to GARCH models

Stochastic Volatility is an alternative class of models that rely on assumption that the log conditional variance follows a non-deterministic autoregressive AR(p) (usually $p = 1$) process (Shapovalova, 2021):

$$ln\sigma_{t+1}^2 = \mu + \phi ln\sigma_t^2 + \sigma_\eta \eta_{t+1}, \tag{7.3}$$

where $\eta_t \sim N(0,1)$ describes the innovation of the log variance process. For the rest of the section we refer the log volatility $ln\sigma_t^2$ as h_t such that $r_t = exp(h_t/2)\varepsilon_t$ where $\varepsilon_t \sim N(0,1)$.

In a multivariate setting, one seeks to simultaneously model the volatility movements of a group of assets (Platanioti et al., 2005). Related movements between different asset classes, financial markets or exchange rates are often observed due to them being infuenced by common unobserved drivers (or factors) (Aydemir, 1998). Diebold and Nerlove (1989) investigated the behaviour of seven dollar exchange rates for a period of 12 years and found that the seven series showed similarities in volatility behaviour in response to actions taken by the US government such as intervention efforts. Since stochastic volatility models are defined in terms of the log volatility process, it is harder to generalise a univariate model to its multivariate counterpart than a GARCH model (Platanioti et al., 2005).

In this experiment I take as baseline the factor model independently proposed by Pitt and Shephard (1999) and Aguilar and West (2000). An open-source package (factorstochvol) was developed in the programming language R by Hosszejni and Kastner (2021) which I used to run the baseline SV model in the analysis. The factor volatility model (Hosszejni and Kastner, 2021) for a portfolio of n assets assumes m latent common factors where $m < n$, it is given that:

$$\begin{aligned} r_t | f_t &\sim \mathcal{N}(\Lambda f_t, \bar{\Sigma}_t), \\ f_t &\sim \mathcal{N}(0, \check{\Sigma}_t), \end{aligned} \tag{7.4}$$

where $f_t = (f_{t1}, ..., f_{tm})^T$ is the vector of m factors, $\Lambda \in \mathbf{R}^{n \times m}$ is a matrix of factor loadings. The covariance matrices $\bar{\Sigma}_t$ and $\check{\Sigma}_t$ are diagonal and are defined as:

$$\begin{aligned} \bar{\Sigma}_t &= diag(exp(\bar{h}_{t1}), ..., exp(\bar{h}_{tn})), \\ \check{\Sigma}_t &= diag(exp(\check{h}_{t1}), ..., exp(\check{h}_{tm})), \end{aligned} \tag{7.5}$$

where $\bar{h}$ and $\check{h}$ are the log variances of the n assets and m latent factors defined as follows (the AR(1) process given in (7.3)):

$$\begin{aligned} \bar{h}_{ti} &\sim \mathcal{N}(\bar{\mu}_i + \bar{\phi}_i(\bar{h}_{t-1,i}), \bar{\sigma}_i^2), i = 1, ..., n, \\ \check{h}_{tj} &\sim \mathcal{N}(\check{\mu}_j + \bar{\phi}_j(\check{h}_{t-1,j}), \check{\sigma}_j^2), j = 1, ..., m. \end{aligned} \tag{7.6}$$

Given the above, the multivariate returns process follows a 0 mean multivariate normal distribution with $r_t \sim \mathcal{N}(0, \Sigma_t)$, where $\Sigma_t = \Lambda \check{\Sigma}_t \Lambda^T + \bar{\Sigma}_t$. It can be seen that the factor volatility model in (7.4) is by nature a state space model with a random walk latent transition process and a linear emission process $r_t | f_t$. I will show later on that the proposed end-to-end neural network architecture follows the same theoretical framework: a neural network (an RNN) that models the non-linear latent transition process $f_t | f_{t-1}$, a neural network (VAE) that infers the latent factors from observational data $f_t | r_{1:t}$, and a neural work (multilayer perceptron (MLP)) that parameterises the emission distribution $r_t | f_t$.

7.3 Designing a purely data-driven deep conditional volatility model

7.3.1 Covariance matrix parameterisation

A covariance matrix is required to be both symmetric and positive definite (Engle and Kroner, 1995). Under the assumption that the returns time series follows a multivariate normal distribution $r_t \sim \mathscr{N}(0, \Sigma_t)$, the aim to evaluate the log determinant $log|\Sigma_t|$ and Mahalanobis distance $r_t^T \Sigma_t^{-1} r_t$ from the log likelihood (7.2). I follow the parameterisation scheme in Dorta et al. (2018) and perform a Cholesky decomposition on the precision matrix:

$$P_t = \Sigma_t^{-1} = L_t L_t^T, \tag{7.7}$$

which ensures P_t is symmetric by construction. To ensure positive definiteness, I require that the diagonal entries of L_t to be strictly positive; this could achieved by applying a Softplus function on top of the neural network output.

It can be seen from the log likelihood (7.2) that the covariance matrix needs to be inverted before evaluating the Mahalanobis distance; this process is costly at higher dimensions. Working with the precision matrix allows one to bypass this inversion during model training as the Mahalanobis distance is simply $r_t^T L_t L_t^T r_t$.

To evaluate the log determinant, it is given that $log|\Sigma_t| = -2\sum_{i=1}^n log(l_{ii,t})$, where $l_{ii,t}$ is the i^{th} element in the diagonal of L_t. Under this scheme, for a portfolio of n assets, the output of the proposed neural network is simply a vector of size $\frac{n(n+1)}{2}$ (denoted z_t thereon). I convert the vector z_t into a lower triangular matrix in a deterministic way using the torch.tril_indices() method in PyTorch (e.g. $f([a,b,c]^T) = \begin{bmatrix} a & 0 \\ b & c \end{bmatrix}$).

I then apply a Softplus function to the diagonal elements of this matrix (to ensure positive definiteness) and the resulting matrix is the lower Cholesky matrix L_t. This procedure is carried out at every time step to produce time-varying precision matrices. When the actual covariance matrix is required, for example in the test set to evaluate model performance, matrix P_t is inverted to obtain Σ_t

7.3.2 Generative model

The generative model defines the joint distribution $P_\theta(r_{1:T}, L_{1:T}, z_{1:T})$, where r_t is the multivariate returns process; L_t is the lower Cholesky decomposition of the precision matrix P_t; vector z_t is the neural network output of size $\frac{n(n+1)}{2}$, which is the latent variable that is to be inferred using observational data. I factorise the joint distribution as follows:

$$P_\theta(r_{1:T}, L_{1:T}, z_{1:T}) = \prod_{t=1}^{T} P_\theta(r_t|L_t)P_\theta(L_t|z_t)P_\theta(z_t|r_{1:t-1}), \tag{7.8}$$

where $P_\theta(z_t|r_{1:t-1})$ is a learned prior distribution which describes the transition dynamics of the latent variable z_t. Information about the sequence $r_{1:t-1}$ is carried by an RNN known as the gated recurrent unit (GRU) (Cho et al., 2014) with hidden state h_t such that:

$$P_\theta(z_t|r_{1:t-1}) = P_\theta(z_t|h_{t-1}) = \mathcal{N}(\mu_{z,t}, \Sigma_{z,t}). \tag{7.9}$$

The prior distribution is parameterised by a multilayer perceptron (MLP):

$$\{\mu_{z,t}, \Sigma_{z,t}\}_{prior} = MLP_{Gen}(h_{t-1}). \tag{7.10}$$

$P_\theta(L_t|z_t)$ is a delta distribution centered on the output of the deterministic function: torch.tril_indices followed by a Softplus function on the diagonal elements, which converts neural network output vector z_t into L_t.

The emission distribution $P_\theta(r_t|L_t)$ (the decoder) describes the 0 mean multivariate normal likelihood given in (7.2) since $P_\theta(r_t|L_t) = P_\theta(r_t|(L_tL_t^T)^{-1} = \Sigma_t)$. A graphical presentation of the generative model is given in Fig 7.1. I refer to the parameters of the generative model collectively as θ, and the parameters of the inference model as ϕ, which are jointly optimised using stochastic gradient variational Bayes.

There are various ways to design the prior distribution $P_\theta(z_t|I_{t-1})$, where

$I_{t-1} = \{r_{1:t-1}, z_{1:t-1}, \Sigma_{1:t-1}\}$ represents all available information up to time $t-1$. I tested other design schemes such as $P_\theta(z_t|r_{1:t-1}, z_{1:t-1})$ and $P_\theta(z_t|r_{1:t-1}, \Sigma_{1:t-1})$, and found that in general the temporal dynamics of the latent variable could be well predicted using past returns alone; hence I decided on $P_\theta(z_t|r_{1:t-1})$. Choosing the prior this way keeps the number of neural network parameters lower than the other two specifications, which reduces overfitting; also one does not need to evaluate the covariance matrix during training.

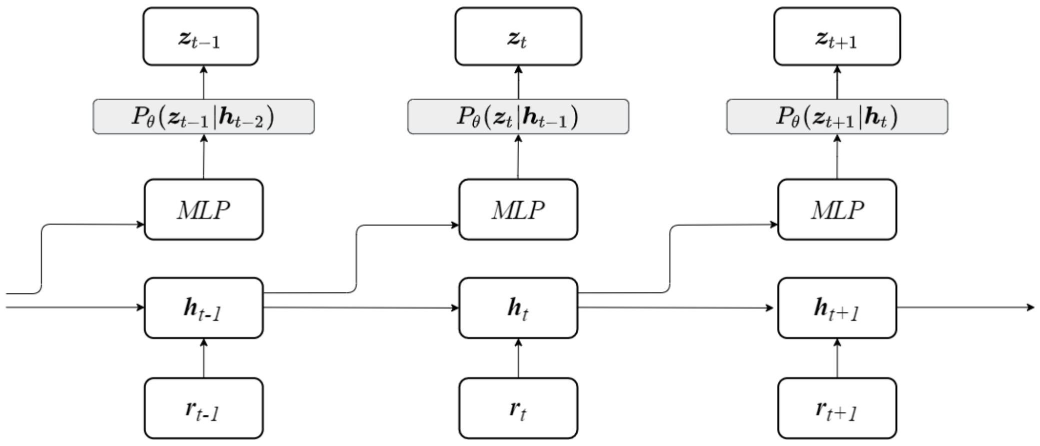


Figure 7.1: Generative model of VHVM. The generative MLP takes as input $\{r_{1:t-1}\}$ and predicts the next-period latent factor z_t, conditioned on which one can obtain an estimate of the covariance matrix Σ_t.

7.3.3 Inference model

The inference model defines the joint distribution $q_\phi(L_{1:T}, z_{1:T}|r_{1:T})$ which I factorise as follows:

$$q_\phi(L_{1:T}, z_{1:T}|r_{1:T}) = \prod_{t=1}^{T} q_\phi(L_t|z_t) q_\phi(z_t|r_{1:t}), \tag{7.11}$$

where the posterior distribution over latent variable $q_\phi(z_t|r_{1:t})$ is parameterised by the encoder (MLP) of the VAE:

$$\{\mu_{z,t}, \Sigma_{z,t}\}_{post} = MLP_{Inf}(h_t); \tag{7.12}$$

this represents the filtering distribution, which is the inference of z_t given the most up-to-date observational data $r_{1:t}$. Since $q_\phi(z_t|r_{1:t})$ is the variational approximation of the actual posterior $P_\theta(z_t|r_{1:t})$, maximising the $ELBO(\theta,\phi)$ is equivalent to minimising the KL divergence between the variational posterior and the actual posterior. The deterministic function to obtain L_t given z_t is the same as in the generative model: i.e. $q_\phi(L_t|z_t) = P_\theta(L_t|z_t)$, since this simply torch.tril_indices() followed by Softplus.

To summarise, VHVM is consisted of three neural networks: (1) an MLP (MLP_{Gen}) for the prior prediction model $P_\theta(z_t|r_{1:t-1})$, also known as the decoder of the VAE which models the transition of the latent variable; (2) an MLP (MLP_{Inf}) for the variational posterior $q_\phi(z_t|r_{1:t})$, which is the encoder the VAE; (3) a GRU with hidden states h_t to carry sequential information about the multivariate returns process $\{r_{1:T}\}$ and is shared by the generative and inference models.

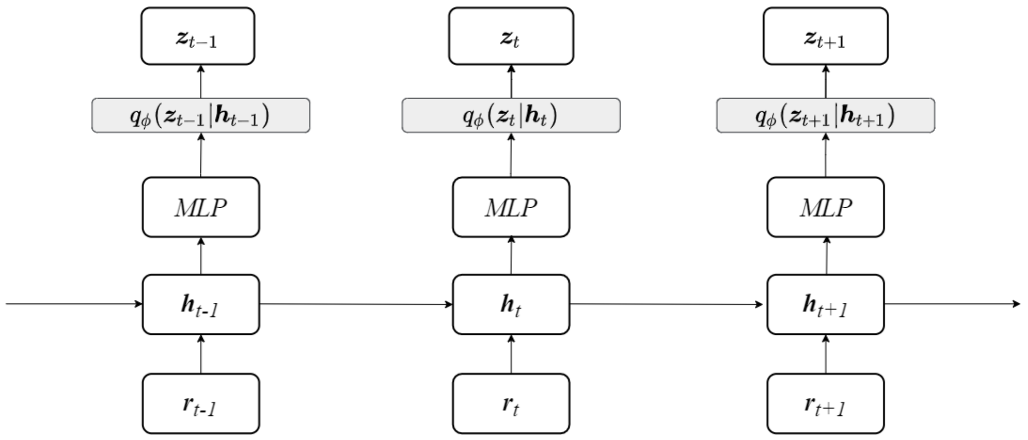


Figure 7.2: Inference model of VHVM. The Inference MLP takes as input $\{r_{1:t}\}$ and paratermises the filter distribution over z_t, conditioned on which one can obtain the posterior estimate of the covariance matrix Σ_t.

7.3.4 Model optimisation and prediction

To perform variational inference I maximise the $ELBO(\theta,\phi)$ w.r.t. θ and ϕ jointly (Kingma and Welling, 2014). The expression for the evidence lower bound is given in (7.13). VHVM is designed to output one step-ahead volatility prediction. When new observations become available, I update the hidden state h_t of the GRU, which serves as the input to the prediction network MLP_{Gen} to predict the next period lower Cholesky matrix.

$$ELBO(\theta,\phi) = \sum_{n=1}^{T} \mathbb{E}_{z_t \sim q_\phi}[logP_\theta(r_t|z_t)] - KL(q_\phi(z_t|r_{1:t})||P_\theta(z_t|r_{1:t})), \quad (7.13)$$

7.4 Performance of VHVM on multivariate FX datasets

I test VHVM on foreign exchange data obtained from the Trading Academy website (eatradingacademy.com). The datasets were checked to have less than 20% missing values and were cleaned to remove outliers beyond 1.5 times interquartile range, before being scaled to have 0 mean and unit variance.

I compute daily log returns using data from the period 24/01/2012 to 23/01/2022 (a total of 3653 observations), from which I remove weekend readings where the change in asset price was 0. I constructed various portfolios using the collection of FX series for n = 5, 10, 20, and 50 (exact portfolios given in the appendix). For model construction, I used a train:validation:test ratio of 80:10:10 and training for 50 epoches.

For model benchmarking, I compared VHVM against three benchmarks: (1) the DCC-GARCH model (Engle, 2002), (2) a factor SV model with MCMC sampling (Hosszejni and Kastner, 2021), and (3) Neural Stochastic Volatility model (Luo et al., 2018).

For hyperparameter optimisation, I adopted a grid search approach. The RNN hidden state was chosen from $\{8,16,24,32,64,128\}$, the MLP hidden layer size from $\{1,2,5,8,16,32,64\}$, the MLP hidden layer number from $\{1,2,3,5,10,15,25,35\}$, the latent space dimension from $\{1,2,3,5,10,15,25,35\}$, the learning rate was chosen from $\{0.1,0.01,0.001,0.0001\}$.

The three baselines are representative models from the current approaches to volatility forecasting: GARCH models, SV models, and deep learning based models. I have chosen DCC-GARCH due to its ability to model dynamic conditional correlation between assets; I implemented the model in R using the package ”rmgarch" (Galanos, 2022). For the factor SV model with MCMC sampler (MCMC-SV), I used the recently developed "factorstochvol" package in R (Hosszejni and Kastner, 2021).

The Neural Stochastic Volatility model (NSVM) Luo et al. (2018) is perhaps most relevant to this work since it was also designed under the VRNN framework. NSVM uses four recurrent neural networks to model temporal dynamics: one for the observed returns series r_t and another for the latent factor z_t in the generative model; similarly for the inference model.

In the proposed model however, I attempted to keep the number of model parameters low by using only one RNN but inputting the hidden state at different time steps to perform prediction and inference. Another key difference between NSVM and VHVM is that the output of NSVM is a low-rank approximation of the time-varying covariance matrix, whereas VHVM outputs the full covariance matrix. A low rank approximation may offer faster computations for higher dimensional portfolios, however I show that VHVM is consistently better in terms of performance.

For model evaluation,I perform one step-ahead covariance matrix forecasting on the test set, and following Wu et al. (2013) and Luo et al. (2018), I use the log likelihood (7.2) as the performance metric since it describes the likelihood of the observed data falling under the estimated distribution.

In Table 7.1 to 7.5 I show the performance of VHVM against the three baseline models: NSVM, DCC-GARCH, and MCMC-SV on various 5 dimensional FX portfolios. For every time step I forecast a 5×5 covariance matrix and in the tables I report the cumulative log likelihood of the test set. I have highlighted in bold the best performing model in terms of log likelihood (higher is better).

It can be observed that VHVM performs best in 17 out of the 20 constructed portfolios. The neural network baseline NSVM however performs best in only one of the portfolios. As previously mentioned, the two key difference between VHVM and NSVM are: (1) VHVM uses a single RNN to carry information about $r_{1:t}$ and the hidden state at different time steps is used for forecasting (h_{t-1})/inference (h_t), whereas NSVM uses four separate RNNs to model z_t and r_t in generation and inference; (2) NSVM outputs low rank approximations of the covariance matrix whereas VHVM outputs estimates of the full covariance matrix.

The simpler structure (fewer parameters) of the proposed model has helped to reduce overfitting. The computational complexity of the proposed model is $O(n^2)$ since the neural network is outputting a vector whose size is proportional to the number of terms in the covariance matrix. Whilst this is more computationally expensive than the $O(n)$ parameterisation in NSVM, I show experimentally that this is justified as I compare the log-likelihoods of various financial portfolios using the predicted covariance matrices.

Table 7.1: Log likelihoods of 5 dimensional Euro-denominated portfolios. The best performing model is highlighted in bold; higher log likelihood is better.

FX pairs	VHVM (proposed)	NSVM	DCC-GARCH	MCMC-SV
EURAUD, EURHKD, EURCAD, EURCNY, EURDKK	−1013.489	-1362.564	-1134.183	-1144.132
EURCNY, EURGBP, EURHKD, EURHUF, EURIDR	−1189.944	-1456.841	-1235.496	-1279.841
EURGBP, EURJPY, EURKRW, EURMXN, EURNOK	−1418.791	-1493.457	-1506.990	-1471.722
EURJPY, EURNZD, EURRUB, EURSGD, EURTHUB	−1222.507	-1331.243	-1301.942	-1262.875

Table 7.2: Log likelihoods of 5 dimensional GBP-denominated portfolios. The best performing model is highlighted in bold; higher log likelihood is better.

FX pairs	VHVM	NSVM	DCC-GARCH	MCMC-SV
GBPAUD, GBPBGN, GBPBRL, GBPCAD, GBPCHF	−1156.903	-1564.931	-1328.515	-1300.182
GBPCHF, GBPCNY, GBPDKK, GBPHKD, GBPILS	−898.588	-1571.065	-1047.304	-1076.312
GBPCNY, GBPINR, GBPJPY, GBPMXN, GBPKRW	−1142.915	-1400.454	-1248.320	-1211.203
GBPRUB, GBPSEK, GBPTRY, GBPJPY, GBPCAD	-1639.355	−1628.892	-2969.575	-2900.870

Table 7.3: Log likelihoods of 5 dimensional USD-denominated portfolios. The best performing model is highlighted in bold; higher log likelihood is better.

FX pairs	VHVM	NSVM	DCC-GARCH	MCMC-SV
USDAUD, USDBGN, USDCAD, USDCHF, USDCNY	−1309.623	-1663.140	-1491.194	-1333.658
USDCNY, USDEUR, USDGBP, USDHKD, USDNZD	-1416.474	-1621.547	-1536.093	−1414.045
USDEUR, USDHUF, USDINR, USDJPY, USDNZD	−1237.078	-1461.547	-1306.365	-1293.873
USDGBP, USDJPY, USDKRW, USDMXN, USDTRY	−1609.177	1807.927	-3490.222	-3129.284

To better gauge the relative performances of the four models, I follow Ismail Fawaz et al. (2019) and plot a critical difference (CD) diagram showing the average ranking of the four model in Figure 7.3. Within a CD diagram, two models without a statistically significant difference (s.s.d.) in average ranking are connected with a horizontal line; the absence of such lines in Figure 7.3 indicates that the four models are s.s.d. in performance across the 20 5 dimensional experiments.

According to Figure 7.3 VHVM has the best overall average ranking (1.25), followed by MCMC-SV(2.2), DCC-GARCH(3), and NSVM(3.55). The fact that MCMC-SV performs slightly better than DCC-GARCH is in accordance with claims that SV models are more flexible at modelling heteroscedastic behaviour in financial time series (Shapovalova, 2021).

Table 7.4: Log likelihoods of 5 dimensional CNY-denominated portfolios. The best performing model is highlighted in bold; higher log likelihood is better.

FX pairs	VHVM	NSVM	DCC-GARCH	MCMC-SV
CNYCAD, CNYEUR, CNYGBP, CNYIDR, CNYJPY	−1171.234	-1447.516	-1304.421	-1272.866
CNYKRW, CNYMXN, CNYMYR, CNYRUB, CNYSEK	−1270.803	-1346.751	-1384.705	-1335.663
CNYGBP, CNYJPY, CNYSEK, CNYSGD, CNYTHB	−1236.308	-1427.729	-1307.665	-1283.031
CNYEUR, CNYMXN, CNYCAD, CNYUSD, CNYTHB	-1604.544	-1592.762	−1535.586	-1541.026

Table 7.5: Log likelihoods of 5 dimensional mixed currency portfolios. The best performing model is highlighted in bold; higher log likelihood is better.

FX pairs	VHVM	NSVM	DCC-GARCH	MCMC-SV
EURAUD, GBPCAD, USDCHF, USDCNY, CNYGBP	−1354.807	-1580.274	–1425.861	-1363.748
EURHKD, GBPJPY, USDCHF, CNYRUB, CNYCAD	−1139.711	-1331.203	-1290.668	-1271.675
USDGBP, USDJPY, GBPCHF, CNYSGD, GBPMXN	−1312.835	-1379.446	-1379.766	-1313.195
CNYEUR, CNYGBP, EURKRW, USDINR, GBPRUB	−1041.155	-1247.187	-1172.433	-1146.655

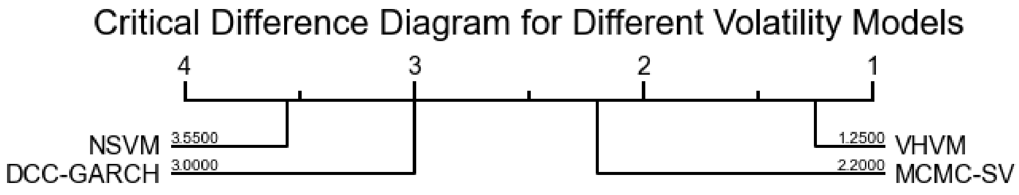


Figure 7.3: Critical difference diagram showing the comparison between VHVM, NSVM, DCC-GARCH, and MCMC-SV in 5 dimensional FX portfolios.

In Table 7.6 I show the experimental results for larger portfolios (10, 20, and 50 dimensions). In these experiments I only compare VHVM and NSVM since both DCC-GARCH and MCMC-SV have difficulties scaling up to higher dimensions. The list of currencies included in each portfolio is included in the Appendix. It can be observed that VHVM performs better than NSVM across all higher dimensional portfolios, consistent with what is seen in the 5 dimensional experiments.

Table 7.6: Log likelihoods of higher dimensional currency portfolios. The best performing model is highlighted in bold; higher log likelihood is better.

FX pairs	VHVM (ours)	NSVM
$10D_1$	-1302.786	-3038.413
$10D_2$	-1804.729	-3117.545
$10D_3$	-1472.019	-3114.550
$10D_4$	-2831.592	-3174.692
$20D_1$	789.993	-6061.243
$20D_2$	-1334.264	-6312.093
$50D$	-2073.592	-15180.516

7.5 Conclusion

In this experiment I have proposed Variational Heteroscedastic Volatility model (VHVM): an end-to-end neural network architecture capable of forecasting one step-ahead covariance matrices.

VHVM outputs the lower Cholesky decomposition of a time-varying conditional precision matrix, which enforces two necessary constraints of a covariance matrix: symmetry and postive definiteness. Furthermore, by setting the neural network to output the precision matrix, one can bypass the computationally expensive matrix inversion step in the evaluation of the multivariate normal log likelihood function.

I demonstrated the effectiveness of VHVM against GARCH, SV, and deep learning baseline models and we observed that VHVM consistently outperformed its competitors.

For potential future extensions of this work, one could explore other types of advanced neural network models such as the transformer to incorporate longer history into the forecasting process. One could also experiment combining natural language processing techniques into volatility forecasting to better explore alternative datasets.

Another line of interesting research is the analysis of the predicted covariance matrix. When the portfolio is of high dimensions, there could be many forecasted correlations between asset pairs that could be attributed to noise. One potential approach is to apply network filtering techniques such as TMFG (Massara et al.,

2016) to denoise the predictions before further analysis.

Chapter 8

General Conclusions

8.1 Summary

The purpose of this thesis is to explore nonlinear time series modelling in the context of latent variable modelling for univariate/multivariate forecasting. This is achieved by adopting the assumption that the underlying temporal dynamics of a time series occur in a latent/unobservable space that is less noisy and contains the true structure of the time series, and an emission dynamics which is a mapping from the latent space to the observation space. This thesis aims to model the transition and emission processes using neural networks to demonstrate the added value of nonlinear modelling.

More specifically, the thesis makes use of two main components: 1. a recurrent neural network to perform nonlinear modelling in the temporal dimension, 2. variational inference to obtain estimates of the latent variables. The combination of the two allows information to propagate between past and present (values of a time series), and between latent and observable spaces. The thesis is consisted of four experiments, and the exact model being proposed in experiment aims to capture this information flow the best way possible in order to learn the inherent structure in the time series to be predicted.

In the introduction section I proposed three research questions for this thesis: 1. can deep learning and latent variable modelling be combined to achieve superior forecasting performance, 2. can deep learning be used to infer relationships between latent and observed time series, 3. how to successfully apply this framework to real world time series. I have addressed these research questions in the following four experiments.

8.1.1 Designing deep latent models for time series forecasting

This experiment proposes a novel architecture to perform probabilistic multistep-ahead time series forecasting with covariates. The model combines a recurrent neural network and a variational autoencoder to perform temporal modelling and inference of latent variables.

The purpose of this experiment is to verify the assumption that univariate/multivariate time series can be projected to a latent space which better describes the temporal evolution of the time series and the relationships amongst a group of time series. Hence, the latent space could be viewed as a set of features which describes the time series of interest and the experiments show that by inferring these features and using them in the forecasting process alongside external covariates, one can achieve better forecasting performance than simply applying an auto-regressive model on the time series. The effectiveness of this setup is verified on a wide range of time series from different domains.

The main advantage of the proposed VRNN model for time series forecasting compared to a vanilla RNN benchmark is the advanced forecasting capability due to representation learning, as well as its generative features and the ability to do probabilistic forecasting. The main drawback is that the representation learning process is a black box and hence the exact features learnt by the variational autoencoder is unknown.

This experiment addresses research question 1 as the proposed neural network latent variable model is able to output satisfactory prediction performance and realistic confidence intervals.

8.1.2 Investigation of inter-time series relationships learnt by the deep latent variable model

This experiment proposes a new model to check for Granger causality between two time series in the presence of a potential confounder. This improves upon most existing Granger causality methodologies which assume the absence of confounding.

From the first experiment I show that there is added benefit in explicitly modelling the mapping from the observation to the latent space, which produces a set of features that can be used for downstream tasks. In this experiment, these features represent a proxy for the potential confounder. I show that for three time series X ,Y and Z, where Z is the confounder, one can obtain better predictions of Y using a proxy for Z (learned by neural networks) than using X, which is the usual way to check for Granger causality when assuming the absence of a confounder.

More specifically, when given X to predict Y, one can use the proposed model to attempt to identify potential confounders, and the use of estimates of these confounders in the forecasting of Y could result in better performance than using X.

The main advantage of the proposed setup is in its explainability. By using other relevant covariates to learn a representation of the confounder, one could use this proxy to compare the forecasting performance of a target series with and without it. This direct comparison allows one to account for the confounder directly, rather than assuming its absence with the causal sufficiency assumption.

The main difficulty of the proposed method lies in determining which covariates can be used to learn the proxy confounder from. At this moment in time, it is assumed that expert judgement is in place to select the relevant time series. For future work, one direction is to propose a more statistical/data driven way to select the covariates.

This experiment addresses research question 2 by showing that nonlinear modelling, through the use of neural networks, allows one to discover complex hidden relationships between observed and unobserved time series. This allows the unobserved to be accounted for explicitly in subsequent analysis.

8.1.3 Designing deep latent conditional volatility models- part 1

This chapter proposes a neural network extension to traditional GARCH models for market volatility forecasting. In the first experiment I show that the autoencoder in the VAE-RNN setup is able to learn useful features about the time series which can be used for prediction, however those features tend to be unexplainable in nature. In this experiment I set the features to be the coefficients of a GARCH model. This essentially creates a GARCH model with time-varying coefficients and I show experimentally that this improves upon traditional GARCH models with constant coefficients and linear modelling.

The main advantage of the proposed approach is that it enhances the predictive capabilities of traditional GARCH models with deep learning architectures, as seen when comparing the performances of neural GARCH with its traditional counterparts. The main drawback is that since neural GARCH are still GARCH models by nature, they suffer from the same problems as traditional GARCH models such as curse of dimensionality and high computational complexity.

8.1.4 Designing deep latent conditional volatility models- part 2

This chapter extends the work of Neural GARCH modelling to higher dimensions. It has been documented in literature that GARCH models can work up to a portfolio of 5 assets. This chapter proposes the use of the VAE-RNN setup to model and predict the covariance matrix of a portfolio up to 50 assets. I introduce a set of constraints to ensure that the neural network outputs a valid covariance matrix and I demonstrate the effectiveness of the proposed model against current SOTA neural network volatility models and traditional baselines such as stochastic volatility models.

This experiment together with Neural GARCH address research question 3 by demonstrating that one can successfully apply deep latent time series models in two ways: 1. combining neural networks with existing econometrics/time models such that their properties also apply to the new model, 2. through the use of external constraints on the neural network output to ensure that their properties align with domain knowledge.

The main advantage of the proposed approach over neural GARCH models is its data driven nature. Since neural GARCH requires expertise in various types of GARCH models, VHVM is an end to end model that leverages representation learning to forecast the covariance matrix given multidimensional returns time series.

8.2 Future work

In this thesis I have explored the use of nonlinear latent variable modelling for univariate/multivariate time series forecasting. I have achieved this by proposing new models which make use of a recurrent neural network and a variational autoencoder. I have demonstrated the effectiveness of this setup on both observable and latent time series prediction. This setup however is not unique, with more advanced sequential models such as the transformer, one can design more sophisticated linkage between past/future and latent observed/latent spaces.

The main application of this thesis has been in the financial domain. It would be interesting to explore similar methodologies in other fields where time series structures are significantly different, such as neurological time series and atmospheric/weather related time series.

Finally, the recent boom of large language models and enhanced generative modelling give rise to potentially interesting intersections between natural language processing and latent variable modelling.

Appendix A

VHVM Neural network hyperparameters

In this section I provide the neural network hyperparameters for the 5D, 10D and 20D VHVM experiments in the following order: RNN hidden state size, multilayer perceptron hidden layer (MLP) size, MLP number of layers, dimension of latent state z, and dimension of portfolio.

FX pairs	hyperparams
EURAUD,EURHKD,EURCAD,EURCNY,EURDKK	16,16,3,4,5
EURCNY,EURGBP,EURHKD,EURHUF,EURIDR	16,16,3,4,5
EURGBP,EURJPY,EURKRW,EURMXN,EURNOK	32,32,3,4,5
EURJPY,EURNZD,EURRUB,EURSGD,EURTHB	32,32,3,4,5

FX pairs	hyperparams
GBPAUD,GBPBGN,GBPBRL,GBPCAD,GBPCHF	32,32,3,4,5
GBPCHF,GBPCNY,GBPDKK,GBPHKD,GBPILS	32,32,3,4,5
GBPCNY,GBPINR,GBPJPY,GBPMXN,GBPKRW	32,32,3,4,5
GBPRUB,GBPSEK,GBPTRY,GBPJPY,GBPCAD	32,32,3,4,5

FX pairs	hyperparams
USDAUD,USDBGN,USDCAD,USDCHF,USDCNY	32,32,3,4,5
USDCNY,USDEUR,USDGBP,USDHKD,USDNZD	32,32,3,4,5
USDEUR,USDHUF,USDINR,USDJPY,USDNZD	32,32,3,4,5
USDGBP,USDJPY,USDKRW,USDMXN,USDTRY	32,32,3,4,5

FX pairs	hyperparams
CNYCAD,CNYEUR,CNYGBP,CNYIDR,CNYJPY	32,32,3,4,5
CNYKRW,CNYMXN,CNYMYR,CNYRUB,CNYSEK	32,32,3,4,5
CNYGBP,CNYJPY,CNYSEK,CNYSGD,CNYTHB	32,32,3,4,5
CNYEUR,CNYMXN,CNYCAD,CNYUSD,CNYTHB	32,32,3,4,5

FX pairs	hyperparams
EURAUD,GBPCAD,USDCHF,USDCNY,CNYGBP	32,32,3,4,5
EURHKD,GBPJPY,USDCHF,CNYRUB,CNYCAD	32,32,3,4,5
USDGBP,USDJPY,GBPCHF,CNYSGD,GBPMXN	32,32,3,4,5
CNYEUR,CNYGBP,EURKRW,USDINR,GBPRUB	32,32,3,4,5

FX pairs	hyperparams
EURAUD,EURHKD,EURCAD,EURCNY,EURDKK GBPAUD,GBPBGN,GBPBRL,GBPCAD,GBPCHF	32,32,3,5,10

FX pairs	hyperparams
EURGBP,EURJPY,EURKRW,EURMXN,EURNOK GBPCHF,GBPCNY,GBPDKK,GBPHKD,GBPILS	32,32,3,5,10

FX pairs	hyperparams
USDAUD,USDBGN,USDCAD,USDCHF,USDCNY CNYCAD,CNYEUR,CNYGBP,CNYIDR,CNYJPY	32,32,3,5,10

FX pairs	hyperparams
USDEUR,USDHUF,USDINR,USDJPY,USDNZD CNYEUR,CNYMXN,CNYCAD,CNYUSD,CNYTHB	32,32,3,5,10

FX pairs	hyperparams
EURAUD,EURHKD,EURCAD,EURCNY,EURDKK GBPAUD,GBPBGN,GBPBRL,GBPCAD,GBPCHF USDAUD,USDBGN,USDCAD,USDCHF,USDCNY CNYCAD,CNYTHB,CNYGBP,CNYIDR,CNYJPY	32,32,3,10,20

FX pairs	hyperparams
EURGBP,EURJPY,EURKRW,EURMXN,EURNOK GBPCHF,GBPCNY,GBPDKK,GBPHKD,GBPILS USDEUR,USDHUF,USDINR,USDJPY,USDNZD CNYEUR,CNYMXN,CNYCAD,CNYUSD,CNYTHB	32,32,3,10,20

FX pairs	hyperparams
CNYAUD,CNYGBN,CNYCAD,CNYDKK,CNYEUR CNYGBP,CNYIDR,CNYINR,CNYJPY,CNYKRW CNYMXN,CNYMYR,CNYNOK,CNYRUB,CNYSEK CNYSGD,CNYTHB,CNYTRY,CNYUSD,CNYZAR EURAUD,EURBGN,EURCAD,EURDKK,EURGBP EURHKD,EURHRK,EURHUF,EURIDR,EURJPY EURKRW,EURMXN,EURMYR,EURNOK,EURNZD GBPAUD,GBPBRL,GBPCAD,GBPCHF,GBPHKD USDAUD,USDCHF,USDCAD,USDHKD,USDHRK USDTRY,USDNZD,USDMXN,USDJPY,USDCZK	64,64,3,25,50

Bibliography

Changing dynamics: Time-varying autoregressive models using generalized additive modeling. *Psychological Methods*, 22(3):409–425, 2017. doi: 10.1037/met0000085.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Zahra Abbasvandi and Ali Motie Nasrabadi. A self-organized recurrent neural network for estimating the effective connectivity and its application to EEG data. *Computers in Biology and Medicine*, 110(May):93–107, 2019. ISSN 18790534. doi: 10.1016/j.compbiomed.2019.05.012. URL `https://doi.org/10.1016/j.compbiomed.2019.05.012`.

Omar Aguilar and Mike West. Bayesian dynamic factor models and portfolio allocation. *Journal of Business and Economic Statistics*, 18(3):338–357, 2000. ISSN 15372707. doi: 10.1080/07350015.2000.10524875.

A.B. Aydemir. *Volatility Modelling in Finance*. Butterworth-Heinemann, Oxford, 1998.

Kasun Bandara, Peibei Shi, Christoph Bergmeir, Hansika Hewamalage, Quoc Tran, and Brian Seaman. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. pages 462–474. Springer International Publishing, 2019. ISBN 978-3-030-36718-3.

Luc Bauwens, Sébastien Laurent, and Jeroen V.K. Rombouts. Multivariate GARCH models: A survey. *Journal of Applied Econometrics*, 21(1):79–109, 2006.

Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint*, pages 1–9, 2014. URL `http://arxiv.org/abs/1411.7610`.

Carolina Bento. Multilayer perceptron explained with a real-life example and python code: Sentiment analysis, 9 2021.

Boris P. Bezruchko, Vladimir I. Ponomarenko, Mikhail D. Prokhorov, Dmitrii A. Smirnov, and Peter A. Tass. Modeling nonlinear oscillatory systems and diagnostics of coupling between them using chaotic time series analysis: applications in neurophysiology. *Physics-Uspekhi*, 51(3):304–310, 2008. doi: 10.1070/pu2008v051n03abeh006494. URL `https://doi.org/10.1070/pu2008v051n03abeh006494`.

Ioana Bica, Ahmed M. Alaa, and Mihaela Van Der Schaar. Time series deconfounder: Estimating treatment effects over time in the presence of hidden confounders. In *37th International Conference on Machine Learning, ICML 2020*, volume PartF16814, pages 861–872, 2020. ISBN 9781713821120.

Tim Bollerslev. Generalised Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31:307–327, 1986.

Tim Bollerslev. A Conditionally Heteroskedastic Time Series Model for Speculative

Prices and Rates of Return. *The Review of Economics and Statistics*, 69(3):542–547, 1987.

Tim Bollerslev, Robert F Engle, and Jeffrey M Wooldridge. A Capital Asset Pricing Model with Time-Varying Covariances. *Journal of Political Economy*, 96(1): 116–131, 1988.

Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang Wang. Probabilistic demand forecasting at scale. *Proc. VLDB Endow.*, 10(12): 1694–1705, aug 2017. doi: 10.14778/3137765.3137775.

Eoin Brophy, Zhengwei Wang, Qi She, and Tomás Ward. Generative adversarial networks in time series: A systematic literature review. *ACM Comput. Surv.*, 55(10), feb 2023. ISSN 0360-0300. doi: 10.1145/3559540. URL `https://doi.org/10.1145/3559540`.

Edward De Brouwer, Adam Arany, Jaak Simm, and Yves Moreau. Inferring causal dependencies between chaotic dynamical systems from sporadic time series. 2020.

Bart Bussmann, Jannes Nys, and Steven Latré. Neural Additive Vector Autoregression Models for Causal Discovery in Time Series Data. *arXiv preprint*, 2020. URL `http://arxiv.org/abs/2010.09429`.

Junyi Chai, Hao Zeng, Anming Li, and Eric W.T. Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134, 2021. ISSN 2666-8270. doi: https://doi.org/10.1016/j.mlwa.2021.100134. URL `https://www.sciencedirect.com/science/article/pii/S2666827021000670`.

Joshua C.C. Chan and Angelia L. Grant. Modeling energy price dynamics: GARCH versus stochastic volatility. *Energy Economics*, 54:182–189, 2016. doi: 10.1016/j.eneco.2015.12.003.

Kinder Chen. Introduction to lstm and gru, 4 2021.

Yonghong Chen, Steven L. Bressler, and Mingzhou Ding. Frequency decomposition of conditional Granger causality and application to multivariate neural field potential data. *Journal of Neuroscience Methods*, 150(2):228–237, 2006. ISSN 01650270. doi: 10.1016/j.jneumeth.2005.06.011.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. *IEEE Access*, 9:120043–120065, 2021. doi: 10.1109/ACCESS.2021.3107975.

George A. Christodoulakis and Stephen E. Satchell. Correlated ARCH (CorrARCH): Modelling the time-varying conditional correlation between financial asset returns. *European Journal of Operational Research*, 139(2):351–370, 2002. doi: 10.1016/S0377-2217(01)00361-7.

Peter F Christoffersen and Francis X Diebold. How Relevant is Volatility Forecasting for Financial Risk Management ? *The Review of Economics and Statistics*, 82(1):12–22, 2000.

Jeffrey Chu, Stephen Chan, Saralees Nadarajah, and Joerg Osterrieder. GARCH Modelling of Cryptocurrencies. *Journal of Risk and Financial Management*, 10 (4):17, 2017. doi: 10.3390/jrfm10040017.

Tianjiao Chu and Clark Glymour. Search for additive nonlinear time series causal models. *Journal of Machine Learning Research*, 9:967–991, 2008. ISSN 15324435.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. 12 2014. URL `http://arxiv.org/abs/1412.3555`.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. volume 2015-Janua, pages 2980–2988, 2015.

Francis X. Diebold and Marc Nerlove. The dynamics of exchange rate volatility: A multivariate latent factor ARCH model. *Journal of Applied Econometrics*, 4(1): 1–21, 1989. ISSN 10991255. doi: 10.1002/jae.3950040102.

Guanqun Dong, Kamaladdin Fataliyev, and Lipo Wang. One-step and multi-step ahead stock prediction using backpropagation neural networks. pages 1–5, 12 2013. ISBN 978-1-4799-0434-1. doi: 10.1109/ICICS.2013.6782784.

Garoe Dorta, Sara Vicente, Lourdes Agapito, Neill D.F. Campbell, and Ivor Simpson. Structured Uncertainty Prediction Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5477–5485, 2018. doi: 10.1109/CVPR.2018.00574.

Xiaodong Du, Cindy L. Yu, and Dermot J. Hayes. Speculation and volatility spillover in the crude oil and agricultural commodity markets: A Bayesian analysis. *Energy Economics*, 33(3):497–503, 2011. doi: 10.1016/j.eneco.2010.12.015.

Andrea Duggento, Maria Guerrisi, and Nicola Toschi. Echo State Network models for nonlinear Granger causality. *bioRxiv*, pages 1–7, 2019. doi: 10.1101/651679.

Michael Eichler. Causal inference in time series analysis. *Causality: Wiley Series in Probability and Statistics*, pages 6–28, 2012. doi: 10.1002/9781119945710.ch22.

Michael Eichler. Causal inference with multiple time series: Principles and problems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1997), 2013. ISSN 1364503X. doi: 10.1098/rsta.2011.0613.

Robert Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4):987–1007, 1982.

Robert Engle. Dynamic Conditional Correlation. *Journal of Business and Economic Statistics*, 20(3):339–350, 2002.

Robert Engle and Kenneth Kroner. Multivariate Simultaneous Generalized Arch. *Econometric Theory*, 11(1):122–150, 1995.

Robert F. Engle and Andrew J. Patton. What good is a volatility model? *Forecasting Volatility in the Financial Markets*, 1:47–63, 2007. doi: 10.1016/B978-075066942-9.50004-2.

I. Erten, M.B. Murat, and N. Okay. Volatility Spillovers in Emerging Markets During the Global Financial Crisis : Diagonal BEKK Approach. *Munich Personal RePEc Archive*, (56190):1–18, 2012.

Marcos Escobar-Anel, Maximilian Gollart, and Rudi Zagst. Closed-form portfolio optimization under GARCH models. *Operations Research Perspectives*, 9: 100216, 2022. doi: 10.1016/j.orp.2021.100216.

Otto Fabius and Joost R. van Amersfoort. Variational recurrent auto-encoders. pages 1–5, 2015.

Eugene F . Fama. The Behavior of Stock-Market Prices Author (s): Eugene F . Fama Published by : The University of Chicago Press Stable. *The Journal of Business*, 38(1):34–105, 1965.

Lucas Borges Ferreira and Fernando França da Cunha. Multi-step ahead forecasting of daily reference evapotranspiration using deep learning. *Computers and Electronics in Agriculture*, 178:105728, 2020. ISSN 0168-1699. doi: https://doi.org/10.1016/j.compag.2020.105728. URL `https://www.sciencedirect.com/science/article/pii/S0168169920314034`.

Sven Festag, Joachim Denzler, and Cord Spreckelsen. Generative adversarial networks for biomedical time series forecasting and imputation. *Journal of Biomedical Informatics*, 129:104058, 2022. ISSN 1532-0464. doi: https://doi.org/

10.1016/j.jbi.2022.104058. URL `https://www.sciencedirect.com/science/article/pii/S1532046422000740`.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. pages 2207–2215, 2016.

M Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

Alexios Galanos. *rmgarch: Multivariate GARCH models.*, 2022. R package version 1.3-9.

Alexios Galanos and Tobuias Kley. *rugarch: Univariate GARCH Models*, 2022. URL `https://CRAN.R-project.org/package=rugarch`. R package version 1.4-7.

Everette S. Gardner. Exponential smoothing: The state of the art—part ii. *International Journal of Forecasting*, 22(4):637–666, 2006. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2006.03.005. URL `https://www.sciencedirect.com/science/article/pii/S0169207006000392`.

Andreas Gerhardus and Jakob Runge. High-recall causal discovery for autocorrelated time series with latent confounders. In *34th Conference on Neural Information Processing Systems*, number NeurIPS, 2020. URL `http://arxiv.org/abs/2007.01884`.

Lawrence R. Glosten, Ravi Jagannathan, and David E. Runkle. On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *The Journal of Finance*, 48(5):1779–1801, 1993. doi: 10.1111/j.1540-6261.1993.tb05128.x.

Simon Godsill. Particle filtering: the first 25 years and beyond. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7760–7764, 2019. doi: 10.1109/ICASSP.2019.8683411.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

Clive J. W. Granger. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica*, 37(3):424–438, 1969.

Peter R. Hansen and Asger Lunde. A forecast comparison of volatility models: Does anything beat a GARCH(1,1)? *Journal of Applied Econometrics*, 20(7): 873–889, 2005. doi: 10.1002/jae.800.

Andrew C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1990. doi: 10.1017/CBO9781107049994.

Syed Aun Hassan and Farooq Malik. Multivariate GARCH modeling of sector volatility transmission. *Quarterly Review of Economics and Finance*, 47(3):470–480, 2007. doi: 10.1016/j.qref.2006.05.006.

Michael Hauser, Yiwei Fu, Yue Li, Shashi Phoha, and Asok Ray. Probabilistic forecasting of symbol sequences with deep neural networks. In *2017 American Control Conference (ACC)*, pages 3147–3152, 2017. doi: 10.23919/ACC.2017.7963431.

Maria Heracleous. Sample Kurtosis, GARCH-t and the Degrees of Freedom Issue. *EUR Working Papers*, 2007.

Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2020.06.008.

URL `https://www.sciencedirect.com/science/article/pii/S0169207020300996`.

Craig Hiemstra and Jonathan Jones. Testing for linear and nonlinear Granger causality in the stock price-volume relation. *The Journal of Finance*, 49(5):1639–1664, 1994.

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

Darjus Hosszejni and Gregor Kastner. Modeling univariate and multivariate stochastic volatility in R with stochvol and factorstochvol. *Journal of Statistical Software*, 100(12):1–34, 2021. doi: 10.18637/jss.v100.i12.

Zexin Hu, Yiqi Zhao, and Matloob Khushi. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4, 2021. ISSN 2571-5577. doi: 10.3390/asi4010009. URL `https://www.mdpi.com/2571-5577/4/1/9`.

Yiyu Huang, Wenjing Su, and Xiang Li. Comparison of bekk garch and dcc garch models: An empirical study. In Longbing Cao, Jiang Zhong, and Yong Feng, editors, *Advanced Data Mining and Applications*, pages 99–110, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

Shun ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5:185–196, 1993. ISSN 0925-2312. doi: https://doi.org/10.1016/0925-2312(93)90006-O. URL `https://www.sciencedirect.com/science/article/pii/092523129390006O`.

Lazaros Alexios Iliadis, Sotirios P. Sotiroudis, Kostas Kokkinidis, Panagiotis Sarigiannidis, Spiridon Nikolaidis, and Sotirios K. Goudos. Music deep learning: A survey on deep learning methods for music processing. In *2022 11th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4, 2022. doi: 10.1109/MOCAST54814.2022.9837541.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019. doi: 10.1007/s10618-019-00619-1.

Eric Jacouier, Nicholas G. Polson, and Peter E. Rossl. Bayesian analysis of stochastic volatility models. *Journal of Business and Economic Statistics*, 12(4):371–389, 1994. doi: 10.1080/07350015.1994.10524553.

J.Duan. The GARCH option pricing model. *Mathematical Finance*, 5(1):13–32, 1995.

Khalil Jebran, Shihua Chen, Irfan Ullah, and Sultan Sikandar Mirza. Does volatility spillover among stock markets varies from normal to turbulent periods? Evidence from emerging markets of Asia. *Journal of Finance and Data Science*, 3(1-4): 20–30, 2017. doi: 10.1016/j.jfds.2017.06.001.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45, 3 1960. ISSN 0021-9223. doi: 10.1115/1.3662552.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van Der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. pages 1–13, 2017.

Mohd Imran Khan and Rajib Maity. Hybrid deep learning approach for multi-step-ahead prediction for daily maximum temperature and heatwaves. *Theoretical and Applied Climatology*, 149:945–963, 2022. ISSN 1434-4483. doi: 10.1007/s00704-022-04103-7. URL `https://doi.org/10.1007/s00704-022-04103-7`.

Saurabh Khanna and Vincent Y. F. Tan. Economy Statistical Recurrent Units For Inferring Nonlinear Granger Causality. In *8th International Conference on Learning Representations, ICLR 2020*, 2020. URL `http://arxiv.org/abs/1911.09879`.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, 82:3713–3744, 2023. ISSN 1573-7721. doi: 10.1007/s11042-022-13428-4. URL `https://doi.org/10.1007/s11042-022-13428-4`.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 12 2014.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. pages 1–14, 2014.

Irena Koprinska, Dengsong Wu, and Zheng Wang. Convolutional neural networks for energy time series forecasting. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018. doi: 10.1109/IJCNN.2018.8489399.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. 11 2015. URL `http://arxiv.org/abs/1511.05121`.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 2101–2109, 2017.

Oussama Lachiheb and Mohamed Salah Gouider. A hierarchical deep neural network design for stock returns prediction. *Procedia Computer Science*, 126:264–272, 2018. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.07.260. URL `https://www.sciencedirect.com/science/article/pii/S1877050918312365`. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia.

Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. Kalman filter and its application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pages 74–77, 2015. doi: 10.1109/ICINIS.2015.35.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=SJiHXGWAZ`.

Bryan Lim and Stefan Zohren. Time series forecasting with deep learning: A survey. 4 2020. doi: 10.1098/rsta.2020.0209. URL `http://arxiv.org/abs/2004.13408http://dx.doi.org/10.1098/rsta.2020.0209`.

Yuntong Liu, Chunna Zhao, and Yaqun Huang. A combined model for multivariate time series forecasting based on mlp-feedforward attention-lstm. *IEEE Access*, 10:88644–88654, 2022. doi: 10.1109/ACCESS.2022.3192430.

H. Viet Long, H. Bin Jebreen, I. Dassios, and D. Baleanu. On the statistical garch model for managing the risk by employing a fat-tailed distribution in finance. *Symmetry*, 12(10):1–15, 2020. doi: 10.3390/sym12101698.

Christos Louizos, Uri Shalit, Joris Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. *Advances in Neural Information Processing Systems*, 2017-December(Nips):6447–6457, 2017. ISSN 10495258.

Rui Luo, Weinan Zhang, Xiaojun Xu, and Jun Wang. A neural stochastic volatility model. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 6401–6408, 2018.

Ali Khalil Malik. European exchange rate volatility dynamics: An empirical investigation. *Journal of Empirical Finance*, 12:187–215, 2005. ISSN 09275398. doi: 10.1016/j.jempfin.2003.09.004.

Daniel Malinsky and Peter Spirtes. Causal Structure Learning from Multivariate Time Series in Settings with Unmeasured Confounding. In *Proceedings of 2018 ACM SIGKDD Workshop on Causal Disocvery*, number 2010, pages 1–25, 2018.

Ričards Marcinkevičs and Julia E. Vogt. Interpretable Models for Granger Causality Using Self-explaining Neural Networks. In *9th International Conference on Learning Representations, ICLR 2021*, 2021. URL `http://arxiv.org/abs/2101.07600`.

Carlo Mari and Emiliano Mari. Deep learning based regime-switching models of energy commodity prices. *Energy Systems*, 2022. ISSN 18683975. doi: 10.1007/s12667-022-00515-6.

Daniele Marinazzo, Wei Liao, Huafu Chen, and Sebastiano Stramaglia. Nonlinear connectivity by granger causality. *NeuroImage*, 58:330–338, 2011. ISSN 10538119. doi: 10.1016/j.neuroimage.2010.01.099. URL `http://dx.doi.org/10.1016/j.neuroimage.2010.01.099`.

Stephen Marra. Predicting volatility. 12 2015.

Guido Previde Massara, T. Di Matteo, and Tomaso Aste. Network Filtering for Big Data: Triangulated Maximally Filtered Graph. *Journal of Complex Networks*, 5 (2):161–178, 06 2016. ISSN 2051-1310. doi: 10.1093/comnet/cnw015. URL `https://doi.org/10.1093/comnet/cnw015`.

S McNally, J Roche, and S Caton. Predicting the price of bitcoin using machine learning. pages 339–343, 2018. ISBN 2377-5750. doi: 10.1109/PDP2018.2018.00060.

Raha Moraffah, Paras Sheth, Mansooreh Karami, Anchit Bhattacharya, Qianru Wang, Anique Tahir, Adrienne Raglin, and Huan Liu. Causal Inference for Time series Analysis: Problems, Methods and Evaluation. *arXiv preprint*, 2021. URL `http://arxiv.org/abs/2102.05829`.

Loris Nanni, Sheryl Brahnam, Michelangelo Paci, and Stefano Ghidoni. Comparison of different convolutional neural network activation functions and methods for building ensembles for small to midsize medical data sets. *Sensors*, 22, 8 2022. ISSN 14248220. doi: 10.3390/s22166129.

Meike Nauta, Doina Bucur, and Christin Seifert. Causal Discovery with Attention-Based Convolutional Neural Networks. *Machine Learning and Knowledge Extraction*, 1(1):312–340, 2019. doi: 10.3390/make1010019.

Daniel Nelson. Conditional Heteroskedasticity in Asset Returns : A New Approach. *Econometrica*, 59(2):347–370, 1991.

Nguyet Nguyen. Hidden markov model for stock trading. *International Journal of Financial Studies*, 6(2), 2018. ISSN 2227-7072. URL `https://www.mdpi.com/2227-7072/6/2/36`.

Junier B. Oliva, Barnabas Poczos, and Jeff Schneider. The statistical recurrent unit. In *34th International Conference on Machine Learning, ICML 2017*, volume 6, pages 4098–4107, 2017. ISBN 9781510855144.

Alvaro D. Orjuela-Canon, Jan A. Freund, Andres Jutinico, and Alexander Cerquera. Granger Causality Analysis based on Neural Networks Architectures for bivariate cases. *Proceedings of the International Joint Conference on Neural Networks*, pages 1–6, 2020. doi: 10.1109/IJCNN48605.2020.9206977.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-l` `pdf`.

Judea Pearl. On measurement bias in causal inference. *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, UAI 2010*, pages 425–432, 2010.

Jonas Peters, Dominik Janzing, and Bernhard Schlkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017. ISBN 0262037319.

Michael K. Pitt and Neil Shephard. *Time varying covariances: a factor stochastic volatility approach*, pages 547–570. Oxford University Press, Oxford, (edited by j.m. bernardo, j.o. berger, a.p. dawid and a.f.m smith) edition, 1999.

K. Platanioti, E.J. McCoy, and D.A. Stephens. A Review of Stochastic Volatility: Univariate and Multivariate Models. 2005.

Ser-Huang Poon and Clive W. J. Granger. Forecasting Volatility in Financial Markets : A Review. *Journal of Economic Literature*, 41(2):478–539, 2003. URL `http://www.jstor.org/stable/3216966`.

Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8, 7 2009.

Pier Francesco Procacci and Tomaso Aste. Forecasting market states. *Quantitative Finance*, 19(9):1491–1498, 2019. doi: 10.1080/14697688.2019.1622313.

Masoomeh Rahimi, Raheleh Davoodi, and Mohammad Hassan Moradi. Deep fuzzy model for non-linear effective connectivity estimation in the intuition of consciousness correlates. *Biomedical Signal Processing and Control*, 57:101732, 2020. ISSN 17468108. doi: 10.1016/j.bspc.2019.101732. URL `https://doi.org/10.1016/j.bspc.2019.101732`.

Rahuljha. https://towardsdatascience.com/lstm-gradients-b3996e6a0296, 6 2020.

Hassan Ramchoun, Mohammed Amine, Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. Multilayer perceptron: Architecture optimization and training. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4:26, 2016. doi: 10.9781/ijimai.2016.415.

Syama Sundar Rangapuram, Matthias Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. volume 2018-Decem, pages 7785–7794, 2018.

Vladimir Ranković, Mikica Drenovak, Branko Urosevic, and Ranko Jelic. Mean-univariate GARCH VaR portfolio optimization: Actual portfolio approach. *Computers and Operations Research*, 72:83–92, 2016. doi: 10.1016/j.cor.2016.01.014.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958. ISSN 1939-1471. doi: 10.1037/h0042519.

David Salinas, Valentin Flunkert, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. 4 2017. URL `http://arxiv.org/abs/1704.04110`.

Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. 11 2019. URL `http://arxiv.org/abs/1912.05911`.

G.William Schwert. Why does stock market volatility change with time? *The Journal of Finance*, 44(5):1115–1153, 1989.

Matthias Seeger, Syama Rangapuram, Yuyang Wang, David Salinas, Jan Gasthaus, Tim Januschowski, and Valentin Flunkert. Approximate bayesian inference in linear state space models for intermittent demand forecasting at scale. 9 2017. URL `http://arxiv.org/abs/1709.07638`.

Yuliya Shapovalova. Exact and approximate methods for bayesian inference: stochastic volatility case study. *Entropy*, 23(4), 2021. ISSN 10994300. doi: 10.3390/e23040466.

Jingyi Shen and M Omair Shafiq. Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of Big Data*, 7:66, 2020.

ISSN 2196-1115. doi: 10.1186/s40537-020-00333-6. URL https://doi.org/10.1186/s40537-020-00333-6.

Kevin Sheppard, Stanislav Khrapov, Gábor Lipták, mikedeltalima, Rob Capellini, alejandro cermeno, Hugle, esvhd, Snyk bot, Alex Fortin, JPN, Matt Judell, Weiliang Li, Austin Adams, jbrockmendel, M. Rabba, Michael E. Rose, Nikolay Tretyak, Tom Rochette, UNO Leo, Xavier RENE-CORAIL, Xin Du, and syncoding. bashtage/arch: Release 5.2.0, March 2022. URL https://doi.org/10.5281/zenodo.6400724.

Peter Spirtes and Kun Zhang. Causal discovery and inference: concepts and recent methodological advances. *Applied Informatics*, 3(1), 2016. ISSN 2196-0089. doi: 10.1186/s40535-016-0018-x.

Adolf Stips, Diego MacIas, Clare Coughlan, Elisa Garcia-Gorriz, and X. San Liang. On the causal structure between CO2 and global temperature. *Scientific Reports*, 6(February):1–9, 2016. ISSN 20452322. doi: 10.1038/srep21691.

Cătălin Stărică and Clive Granger. Nonstationarities in stock returns. *Review of Economics and Statistics*, 87(3):503–522, 2005. doi: 10.1162/0034653054638274.

Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B. Fox. Neural granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2021. ISSN 19393539. doi: 10.1109/TPAMI.2021.3065601.

Violeta Teodora Trifunov, Maha Shadaydeh, Jakob Runge, Veronika Eyring, Markus Reichstein, and Joachim Denzler. Nonlinear Causal Link Estimation Under Hidden Confounding with an Application to Time Series Anomaly Detection. In *41st DAGM German Conference on Pattern Recognition, DAGM GCPR 2019*, volume 11824 LNCS, pages 261–273, 2019. ISBN 9783030336752. doi: 10.1007/978-3-030-33676-9_18.

Y. K. Tse and Albert K.C. Tsui. A multivariate generalized autoregressive conditional heteroscedasticity model with time-varying correlations. *Journal* *of Business and Economic Statistics*, 20(3):351–362, 2002. doi: 10.1198/073500102288618496.

Torsten Ullrich. On the autoregressive time series model using real and complex analysis. *Forecasting*, 3:716–728, 12 2021. ISSN 25719394. doi: 10.3390/forecast3040044.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016. URL `https://arxiv.org/abs/1609.03499`.

Roy Van Der Weide. GO-GARCH: A multivariate generalized orthogonal GARCH model. *Journal of Applied Econometrics*, 17(5):549–564, 2002. doi: 10.1002/jae.688.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Raul Vicente, Michael Wibral, Michael Lindner, and Gordon Pipa. Transfer entropy-a model-free measure of effective connectivity for the neurosciences. *Journal of Computational Neuroscience*, 30(1):45–67, 2011. ISSN 09295313. doi: 10.1007/s10827-010-0262-3.

Matthew Wang, Yi-Hong Lin, and Ilya Mikhelson. Regime-switching factor investing with hidden markov models. *Journal of Risk and Financial Management*, 13(12), 2020. ISSN 1911-8074. doi: 10.3390/jrfm13120311. URL `https://www.mdpi.com/1911-8074/13/12/311`.

Yueming Wang, Kang Lin, Yu Qi, Qi Lian, Shaozhe Feng, Zhaohui Wu, and Gang Pan. Estimating brain connectivity with varying-length time lags using a recurrent neural network. *IEEE Transactions on Biomedical Engineering*, 65:1953–1963, 2018. ISSN 15582531. doi: 10.1109/TBME.2018.2842769.

Wikipedia. Variational autoencoder.

F Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1 (6):80–83, 1945.

Yue Wu, José Miguel Hernández Lobato, and Zoubin Ghahramani. Dynamic covariance models for multivariate financial time series. In *30th International Conference on Machine Learning, ICML 2013*, volume 28, pages 1595–1603, 2013.

Diethelm Wuertz, Tobias Setz, Yohan Chalabi, Chris Boudt, Pierre Chausse, and Michal Miklovac. *fGarch: Autoregressive Conditional Heteroskedastic Modelling*, 2017. URL `https://CRAN.R-project.org/package=fGarch`. R package version 3042.83.2.

Zexuan Yin. Neural garch datasets, Sep 2021. URL `https://github.com/zy2514/Neural-Generalised-Autoregressive-Conditional-Heteroskedasticity`.

Zexuan Yin and Paolo Barucca. Stochastic recurrent neural network for multistep time series forecasting. In *Neural Information Processing*, pages 14–26, Cham, 2021. Springer International Publishing. ISBN 978-3-030-92185-9.

Zexuan Yin and Paolo Barucca. Deep recurrent modelling of granger causality with latent confounding. *Expert Systems with Applications*, 207, 11 2022. ISSN 09574174. doi: 10.1016/j.eswa.2022.118036.

Alex E. Yuan and Wenying Shou. Data-driven causal analysis of observational time series: A synthesis. *bioRxiv*, 2020. ISSN 26928205. URL `https://doi.org/10.1101/2020.08.03.233692`.

Liu Yunpeng, Hou Di, Bao Junpeng, and Qi Yong. Multi-step ahead time series forecasting for different data patterns based on lstm recurrent neural network. In *2017 14th Web Information Systems and Applications Conference (WISA)*, pages 305–310, 2017. doi: 10.1109/WISA.2017.25.

Mengqi Zhang, Xin Jiang, Zehua Fang, Yue Zeng, and Ke Xu. High-order hidden markov model for trend prediction in financial time series. *Physica A: Statistical Mechanics and its Applications*, 517:1–12, 2019. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2018.10.053. URL `https://www.sciencedirect.com/science/article/pii/S0378437118314018`.

Xun Zheng, Manzil Zaheer, Amr Ahmed, Yuan Wang, Eric P Xing, and Alexander J Smola. State space lstm models with particle mcmc inference. 11 2017. URL `http://arxiv.org/abs/1711.11179`.

Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In Feifei Li, Guoliang Li, Seung-won Hwang, Bin Yao, and Zhenjie Zhang, editors, *Web-Age Information Management*, pages 298–310, Cham, 2014. Springer International Publishing.

Xiao Zhong and David Enke. Predicting the daily return direction of the stock market using hybrid machine learning algorithms. *Financial Innovation*, 5:24, 2019. ISSN 2199-4730. doi: 10.1186/s40854-019-0138-0. URL `https://doi.org/10.1186/s40854-019-0138-0`.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021. URL `www.aaai.org`.