On Explainable Deep Learning for Macroeconomic Forecasting and Finance

Cosimo Izzo

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Faculty of Engineering Sciences

UCL Institute of Finance & Technology

University College London

December 26, 2022

I, Cosimo Izzo, 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.

To my father. You are my source of inspiration.

Abstract

Deep Learning (DL) has gained momentum in recent years due to its incredible generalisation performance achieved across many learning tasks. Nevertheless, practitioners and academics have sometime been reluctant to apply these models because perceived as *black boxes*. This is particularly problematic in Economics and Finance. The objective of this thesis is to develop interpretable DL models and explainable DL tools with a focus on macroeconomic and financial applications. In doing so we highlight connections between such models and the standard economic ones.

The first part of this work introduces a new class of interpretable models called Deep Dynamic Factor Models. The study merges the DL literature on autoencoders with that of the Econometrics on Dynamic Factor Models. Empirical validations of the approach are carried out both on synthetic and on real-time macroeconomic data.

Part two of the work analyses feature attribution methods and Shapley values among explainability tools that are used to additively decompose model predictions. One of their limitations is highlighted, given that it is necessary to define a baseline that represents the *missingness* of a feature. A solution to the problem is proposed and compared against the ones currently in use both on simulated data and in the financial context of credit card default. We show that the proposed baseline is the only one that accounts for the specific use of the model.

The final part of the work discusses the use of DL techniques for dynamic asset allocation. Using US market data, a comparison in recursive out-of-

sample among different machine learning, economic-financial and hybrid models, including the one introduced in the first part of the work, is performed. Finally, a nonlinear factor-based portfolio performance attribution via the use of Shapley values and the baseline proposed in part two of the work is presented.

Impact Statement

Deep Learning has increasingly been applied in the fields of Economics and Finance. Yet, sometimes academics and practitioners in such fields are reluctant to the adoption of deep learning methods because they are perceived as *black boxes*. Meaning that for the user it is difficult if not impossible to understand why the model is making a given prediction, suggesting a given choice or behaving in a certain way. There can be cases where this is not relevant, but it certainly is for economists, being the model usually deployed for policy making, or investment decisions, among others. The aim of this PhD thesis is to assist in alleviating this problem by building interpretable models and explainability tools.

In Chapter 4 a new class of models called Deep Dynamic Factor Models is introduced. The approach merges the econometric literature on Dynamic Factor Models with the deep learning one on autoeconders. Chapter 4 is based on a joint work ('Deep Dynamic Factor Models') with Paolo Andreini and Giovanni Ricco, and was accepted for presentation at the following conferences: 2^{nd} Vienna Workshop on Economic Forecasting 2020, 40^{th} International Symposium on Forecasting (ISF 2020).

In Chapter 5 we present a new baseline for Shapley based feature attribution methods, which is shown to outperform other commonly adopted baselines in the empirical evaluations. This chapter is grounded on a joint work ('A Baseline for Shapley Values in MLPs: from Missingness to Neutrality') with Aldo Lipani, Francesca Medda and Ramin Okhrati, and was accepted as a conference paper for the 29^{th} European Symposium on Artificial Neural Networks, Computational

Intelligence and Machine Learning (ESANN 2021).

Finally, in Chapter 6 we discuss the potential of designing financial theory driven reward function that can be optimised with deep learning methods for dynamic portfolio optimisation. Moreover, this chapter introduces a methodology to perform nonlinear factor based portfolio performance attribution through the techniques developed in Chapter 5. Chapter 6 has been inspired by the Financial Analytics and Machine Learning, and the Asset Pricing modules.

The target audience for this PhD thesis stretches beyond the academic field. Indeed, we hope the methods discussed here will be helpful both to practitioners and academics alike. In particular, the content of Chapter 4 can be useful to both public and private institutions to assess macroeconomic conditions, as it is currently done with Dynamic Factor Models, one of the main work-horses for macro-econometric analysis. In fact, the modelling approach proposed is a natural extension of those. The contribution of Chapter 5 goes beyond the economic audience, as the approach when applicable can be useful as a general explainability tool to justify model based decision making. The findings of Chapter 6 support the potential of hybrid strategies merging financial theory with machine learning for investing. Although further research and experiments are unequivocally needed, these results are encouraging.

Acknowledgements

This PhD thesis is the result of numerous interactions with many people, and it is now time for me to express my sincere gratitude to everyone who helped make this work come to fruition.

First, I would like to thank my supervisors, Prof. Francesca Medda, Dr. Aldo Lipani, and Dr. Ramin Okhrati for their priceless guidance. Without them, this work would not have been possible. I am also thankful to Dr. Paolo Andreini, Dr. Fabrizio Coiai, Dr. Jan Hannes Lang, Mr. Filippo Pellegrino, Dr. Alicia Reyes Revuelta, Prof. Giovanni Ricco, Mr. Kevin Sinani, Prof. Chak Wong, the UCL Institute of Finance and Technology and Now-casting Economics for their invaluable advice and support.

I am thankful to the anonymous reviewers who helped to improve some parts of this PhD thesis.

I am also thankful to my Professors in Bocconi, who introduced me to the amazing science of Economics. It is there where all of this started.

I am deeply thankful to my father Sergio; he has been the main source of inspiration and motivation throughout my life, and this PhD is the result of his advice to me about always following my passion. I would like also to thank my dear mother Maria Cristina, my grandmother Giovanna, my sisters Silvia and Giovanna, and my darling Federica. Their support has been of paramount importance for the completion of this PhD, and it will always remain for whatever challenges life offers me.

Finally, I am indebted to all my family and friends.

Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Many economic and financial institutions have been recently investing in Data Science and Machine Learning, ranging from central banks and hedge funds to investment and retail banks, and to contemporary financial technology [1, 2, 3, 4]. Current applications include portfolio optimisation, algorithmic trading and hedging, asset and derivative pricing, financial risk and risk management, factor investing, sentiment analysis, forecasting of economic aggregates, among many others.

However, Artificial Intelligence (AI) and Machine Learning (ML) research in Economics and Finance are not new, occurring at least since 1986 [4]. Indeed, significant attempts to introduce machine learning methods in general, and neural networks in particular, in Economics date back to the 90's with works from *White* [see 5, 6, 7, 8, for example]. Nevertheless, and as discussed in *Athey* [9], at that time they did not lead to significant performance improvements, and thus they did not become popular among economists. More recently, thanks to significant innovations in the field, and the availability of new more powerful computational resources and that of open source libraries (e.g., TensorFlow, PyTorch), Machine Learning is experiencing increasing adoptions in Economics and Finance, as documented in *Godell et al.* [4] with respect to financial applications, in *Doerr et al.* [3] for big data in central banking, in *Nosratabadi*

et al. [1] when it comes to the economic sphere, in *Huang et al.* [2] for Finance and Banking, and in *Athey* [9] for general applications to Economics.

There is however a fundamental difference between Machine Learning and Econometrics.[1] Indeed, and as discussed in *Athey and Imbens* [11], machine learning methods typically rely on data-driven model selection, usually through a method called cross-validation.[2] Therefore, often these models miss formal properties, which are the focus of the economic literature. There are however exceptions [see 12, for a recent example]. That being said, and as discussed in *Mullainathan and Spiess* [13] and in *Athey and Imbens* [14], there are many problems in Economics and Finance that are primarily concerned with the prediction issue. In these cases, selecting a model based on the prediction performance may be sufficient. Moreover, a key argument in favour of applying machine learning techniques to macroeconomic and financial data is their ability to effectively capture nonlinearities [15, 16].

Indeed, some recent works have successfully applied these techniques to such data. *Cook and Hall* [17] employed a number of neural network architectures, including also autoencoders, to forecast the U.S. unemployment rate. *Loermann and Maas* [18] proposed a neural net model to predict the U.S. GDP. *Holopainen and Sarlin* [19] carried out a horse race among different machine learning methods and showed that such models are able to outperform conventional statistical approaches when predicting crisis periods. Additionally, other authors employ Machine Learning (ML) and Deep Learning (DL) on rich datasets incorporating both stock data and macroeconomic aggregates to

[1]The most common tools used to absorb economic and financial data can be found in the field of Econometrics, whose official birth can be attributed to the first issue of Econometrica in January 1933 by Ragnar Frisch[10], where Econometrics is conceived as the unification of three view points, that of statistics, economic theory, and mathematics. Since then Econometrics has played a crucial rule in Economics and Finance which has been growing over time. Witnesses of this are the many Nobel Prizes in Economics given to econometricians (e.g., Ragnar Frisch, 1969; Lawrence Klein, 1980; Trygve Haavelmo, 1989; James Heckman and Daniel McFadden, 2000; Robert Engle and Clive Granger, 2003; Christopher Sims, 2011; Lars Hansen, 2013).

[2]Cross-validation consists in a resampling approach that uses different data for training and validating different models. The final model is selected based on the best performance metric computed on the so called validation set(s).

predict stock returns [20, 21, 22]. Finally, a number of papers have proposed the use of deep learning techniques in order to solve the asset allocation problem [see 23, 24, 25, 26, 27, for example]. These are only few of the works that have shown promising performances of machine learning approaches in general, and deep learning ones in particular.

1.2 Problem Statement and Aims of the PhD Thesis

There is however a major impediment to machine learning deployments to the field of Economics and Finance, and it relates to their *black box* nature. Following *Rudin* [28], a black box model is defined as either a function that is too complicated for a human to comprehend (e.g., deep learning models) or a function that is proprietary. In this study we are concerned with models of the first kind and this problematic goes beyond the application to economic and financial data, and motivated the birth of a new branch of AI called eXplainable AI (XAI)[29], which has seen growing interest in recent years [30]. In particular, in *Gunning* [29] XAI is considered essential to understand, trust, and effectively manage the emerging generation of artificial intelligence.

Following this literature and with a focus on economic and financial problems, this PhD thesis intends to focus on the study and the design of models that, while being able to outperform state of the art econometric competitors, maintain interpretable aspects. When we find not possible to achieve this, we develop explainability tools that are in line with how the human user could adopt the deep learning model. On the empirical side, our focus will be on macroeconomic forecasting and financial applications, the two are intrinsically connected, as any financial decision should take into consideration an assessment of the surrounding macroeconomic conditions.

1.3 Contributions

The contributions of this thesis are:

- The introduction of a new modelling framework that combines deep learning techniques with econometric ones. The framework is a natural extension of Dynamic Factor Models (DFMs), thus able to handle missing data and mixed frequencies in a unique coherent manner. Further, it covers the more general as it can be conceptualized as an effective way to estimate nonlinear state space models with a factor structure;

- The introduction of a novel approach that improves Shapley-based explainability techniques for deep learning models by proposing an alternative way to define baselines for Shapley values. The method allows for an interpretation of the result where feature attributions are computed with respect to a meaningful value for the model, that is a point of the model's maximum uncertainty with respect to taking a particular decision. The advantage of such approach are shown both on synthetic data and in the financial context of credit card default classification;

- An assessment of how the aforementioned can be used and combined with other researches from DL, Finance and Economics, to deliver to hybrid methodologies that can be applied to asset allocation. A theoretical framework based on recent works about Deep Learning applications to portfolio optimisation is proposed. The framework covers both portfolio construction and performance attribution with respect to financial-economic risk factors [31, 32, 33, 34, 35].

1.4 Structure of the Thesis

The thesis is organised such that:

Chapter 2 discusses relevant works that relate to this thesis. We start our review with general studies regarding the application of machine learning models to Economics and Finance. We then narrow down our focus to one of the empirical analysis of this thesis, the problem of macroeconomic forecasting. Here we survey both standard economic approaches from the literature, and more recent ML ones. We then discuss in more details the problem of portfolio

optimisation and review current ML applications. We conclude this chapter by surveying recent explainable AI (XAI) studies in the context of Economics and Finance.

Chapter 3 provides a more formal methodological background to this thesis which will then form the basis of subsequent chapters. Here we formally introduce Dynamic Factor Models, deep learning models and the portfolio optimisation problem. We then formalise the explainability problem in Artificial Intelligence, and discuss feature attribution methods for Deep Learning.

Chapter 4 introduces a new modelling framework that merges the deep learning literature on autoencoders with the econometric one on Dynamic Factor Models (DFMs). The framework is a natural extension of DFMs, thus able to handle missing data and mixed frequencies in a unique coherent manner. Further, it covers the more general as it can be conceptualised as an effective way to estimate nonlinear state space models with a factor structure. The chapter discusses theoretical foundations of the approach, while corroborating them on synthetic and real data. These empirical evaluations are carried out against DFMs with well studied statistical properties [see 36, 37, for example] and currently used as state-of-the art benchmarks for macroeconomic forecasting. The framework proposed maintains aspects of interpretability common to Dynamic Factor Models; e.g., the construction of coincident indicators [38] and the computation of the so called *news* and *impact* [39].

Chapter 5 introduces a novel methodology which, combined with currently available explainability methods in the context of feature attribution, delivers to an explanation of the prediction which takes into account how the model is used. The method allows for an interpretation of the result where feature attributions are computed with respect to a meaningful value for the model, that is a point of the model's maximum uncertainty with respect to taking[3] a particular decision. Empirical evaluations are carried out both on a synthetic dataset and on a financial one about defaults of credit cards.

[3]or suggesting, depending on the context;

Chapter 6 discusses the potential of combining Deep Learning and portfolio choice theory. A horse race in the form of a recursive out-of-sample comparison on US stock market data among different machine learning, economic-financial and hybrid models, including the use of the model introduced in Chapter 4 is performed. Finally, a methodology to compute nonlinear factor-based portfolio performance attributions, via the use of Shapley values and the baseline proposed in Chapter 5, is presented.

Chapter 7 summarises the main findings, sets out the limitations of the studies, and outlines future work.

Finally, in Appendix A, B and C additional details and experiments are provided.

Chapter 2

Literature Review

In Section 2.1 we survey machine learning (ML) applications in Economics and Finance. As the forecasting of such data is among the main topics of this thesis, a particular emphasis is devoted to works that analyse it, although the literature is vastly dominated by financial applications. In Section 2.2 we discuss the specific problem of macroeconomic forecasting both via standard economic approaches and through the newer ones adopting ML tools. In Section 2.3 we carry out a survey of works that are related to the portfolio optimisation problem, both the old ones that initiated the statistical treatment of the problem and the new ones that adopt ML. Finally, in Section 2.4 a review of the fewer literature available in the emerging area of explainability in ML for Economics and Finance is presented.

2.1 Machine Learning in Economics and Finance

Many economic and financial institutions have been recently investing in Data Science and Machine Learning, ranging from central banks and hedge funds to investment and retail banks, and to contemporary financial technology [1, 2, 3, 4]. Although machine learning tools have gained significant interest in Economics and Finance over traditional econometric models, there is still great

potential for development.[1] Current applications include portfolio optimisation [41, 42], algorithmic trading and hedging [43, 44], asset and derivative pricing [45, 46], financial risk and risk management [47, 48], factor investing [22], bond risk predictability [16], sentiment analysis [49, 50, 51], trade settlements [52], automation [53], forecasting of economic aggregates [17, 18]. Credit risk and bankruptcy with hybrid econometric and machine learning models are analysed in *Prado* [54]. *Elliott and Timmermann* [55] discuss forecasting methods in Economics and Finance and conclude that no single model can be expected to dominate across different economic variables and time periods. A key argument in favour of applying ML techniques to economic and financial data is their ability to capture nonlinearities, which stands in contrast to the standard tools used for such data, as discussed in *De Prado* [15] and *Bianchi et al.* [16], for example.

With respect to the forecasting of currency exchange rates, *Nag and Mitra* [56] show the superior performance of hybrid machine learning techniques, compared to classical methods. In *Arroyo et al.* [57] different traditional and new methods are compared to forecast volatility in interval time series. Deep belief networks are observed to outperform autoregressive moving average, the random walk and multilayer perceptrons on the exchange rate problem in *Shen et al.* [58]. *Ravi et al.* [59] discuss the potential of a hybrid machine learning approach to the problem, including chaos theory and evolutionary algorithms. In *Galeshchuk and Mukherjee* [60] the potential of Convolutional Neural Network is shown with respect to the sign change prediction problem.

Stock Market has attracted the highest attention among practitioners and academics [1]. In *Chen et al.* [61] different algorithms are compared on the Chinese stock market, including a combination of an autoencoder and a Random Boltzmann Machine. *Singh and Srivastava* [62] consider Google data and the NASDAQ index and show that a hybrid 2-directional and 2-dimensional Principal Component Analysis augmented with a Deep Neural

[1]For example, in *Ferreira et al.* [40] it has been documented that 90% of the operations carried out in the hedge found industry are performed by a hardcoded procedure.

Network can outperform Radial Basis Function Neural Network and Recurrent Neural Network. *Hernandez and Abad* [63] analyse the performance of different models for the prediction of the sign direction of the NASDAQ-100 index. *Fischer and Krauss* [64] show successful applications of Long Short Term Memory (LSTM) type of networks in the prediction of the directional movements of the constituents of the S&P 500 index. They show that such type of models can outperform memory-free classification methods like the Random Forest, simple Deep Neural Networks, and the Logistic regression. Convolutional Neural Networks and order-based features are used in *Tashiro et al.* [65] to predict stock market mid-price trends.

In *Abe and Nakayama* [66] 25 risk factors are used as inputs to different machine learning models to predict the one-month-ahead stock returns in the cross-section of the Japanese stock market. They show that deep neural networks generally outperform shallow neural networks and other machine learning models. More complicated pipelines are developed in *Bao et al.* [67] where the authors propose a combination of Wavelet Transforms, Stacked Autoencoders, and Long Short Term Memory (LSTM) to predict the stock price. In the first stage, Wavelet Transform eliminates noise by decomposing the stock price time series, then predictive features for the stock price are created via autoencoders. Finally, the LSTM architecture is applied to predict the next day's closing price based on the previous stage's features. Similarly, *Yan and Ouyang* [68] combine wavelet analysis and LSTM to forecast the daily closing price of the Shanghai Composite Index. *Chatzis et al.* [69] show that deep neural networks can outperform other machine learning models in terms of classification accuracy for market crisis predictions.

The use of news data has also received certain attention and has been shown to be beneficial to the forecasting problem. In *Krausa and Feuerriegel* [70] a sentiment indicator via word embedding is considered, and the potential of transfer learning together with deep neural networks is shown with respect to the prediction of stock market directions in response to financial disclosure.

Matsubara et al. [71] combine news data and deep neural generative models to predict daily stock price movements of the Nikkei 225 and Standard & Poor's 500 indices. They compare this framework to multilayer perceptron and supported vector machines. *Minh et al.* [72] combine historical price data and news articles together with the Harvard IV-4 dataset to predict price directions of the Standard & Poor's 500 index and the VN-index via different types of neural networks, among which recurrent neural networks show better performance. In *Becker and Reinganum* [73] an overview of big data and sentiment analysis applications to the problem is presented. The authors highlight techniques such as Natural Language processing, Supported Vector Machines and Artificial Neural Networks for the analysis of news, conference calls, reports, and social media activity. According to their research, sentiment information has provided short-term usefulness, but little long-term insights.

In the sphere of machine learning applications to derivative products, *Hsu et al.* [74] propose a hybrid Black-Scholes and fully connected multilayer perceptron to predict the bid-ask spread of option prices. In *Buehler et al.* [44] a deep learning framework to hedge portfolio of derivatives that takes into account market frictions such as transaction costs, market impact, liquidity constraints or risk limits, is presented. With respect to the responsible investment sphere, *Vo et al.* [75] apply multivariate bidirectional LSTM.

2.2 Macroeconomic Forecasting and Machine Learning

It is possible to distinguish between two antipodal approaches to the problem of macroeconomic forecasting. One that aims to be fully grounded on economic theory, where aggregate relations emerge from the behaviour of interconnected micro-founded agents. We can identify such approach in Dynamic Stochastic General Equilibrium models (DSGE). Following *Smets et al.* [76], those models are characterised by the derivation of behavioural relationships from the optimising behaviour of agents subject to different constraints and the specification

of a well-defined general equilibrium. Their estimation on macroeconomic data can be carried out in the framework of Bayesian likelihood estimation, where prior information about the structural economic parameters can be postulated in a relatively easy manner.

An early successful application of such models to forecasting has been shown by *Smets and Wouters* [77], where a DSGE with real and nominal frictions estimated with Bayesian methods is compared to Bayesian and frequentist vector autoregressive models (VARs) in an out of sample forecasting exercise on US data. The model is shown to achieve competing performance. Nowadays, more complex versions of these models including the modelling of financial frictions [78, 79, 80] are among the standard tools for forecasting, conditional forecasting and scenario analysis, structural and policy analysis by central banks. However, the usefulness of economic theory for forecasting has been sometimes provocatively and rightfully challenged. For example, *Giacomini* [81] makes a critical assessment of whether economic theory can help in forecasting key macroeconomic variables, and found many DSGE models to be outperformed by other approaches. However, DSGE are not the only theory based approaches, and the author finds other theoretical grounded information to be more useful, such as accounting identities, disaggregation and spatial restrictions, and cointegrating relationships. This of course does not mean that DSGE are not useful for forecasting, also because the field is vibrant, and new models are being developed continuously, together with alternative estimation methods [see 80, 82, for example].

On a complete opposite side there are the purely statistical approaches, such as reduced form vector autoregressive models. The econometric literature has then provided ways to impose restrictions driven by economic theory in order to recover a structural representation of the reduced form model. DFMs are statistical models which impose restrictions that are particularly well suited to capture some empirical features of economic time series. These models were firstly introduced by *Geweke* [83] and *Sargent and Sims* [84] and are the very

first instance of *big data* in the discipline. DFMs deal with a large cross-section problem by applying a linear dynamic latent state framework to the analysis of economic time series. The underlying assumption of these models is that there is a small number of pervasive, unobserved, common factors that stir the economy and inform the comovements of hundreds of economic variables. Economic times series are also affected by variable-specific (idiosyncratic) disturbances. These idiosyncratic disturbances can be due either to measurement error or to factors that are specific to some variables. A large body of empirical evidence produced using dynamic factor models, shows how a small number of factors – as many as two – can capture a dominant share of the variance of all the key macroeconomic and financial variables. This family of models has been applied intensively in Econometrics to different problems such as forecasting, structural analysis and the construction of economic activity indicators [see 85, 86, 87, 88, 89, 90, 91, 92, among others], and it is a major work horse in the context of macroeconomic forecasting especially at extremely short horizon, in the so called prediction of the present and the very recent past. This is relevant for macroeconomic data which are published with a lag and subject to revision. The forecasting at these three horizons takes the name of *Nowcasting*, as discussed in *Bańbura et al.* [93]. While the model parameters are usually estimated under frequentist lens, a full Bayesian maximum likelihood approach is possible as well [see 94, 95, 96, 97, 98, 99, 100, for example].

At a first glance DFMs and DSGE could seem unrelated, but they both actually have a state-space model formulation. In particular, DFMs and log-linearised DSGE[2] can be both formulated such that the learning process of the parameters is computed in terms of the likelihood of a linear Gaussian State Space model with a factor structure.[3] They however differ in the restrictions

[2]The system of equations derived from the DSGE are nonlinear and are usually log-linearised around the steady states to facilitate the estimation. This together with the assumption of Gaussian innovations allows to convert the model to a linear Gaussian state space.

[3]These models do not need to be parametric, and non-parametric versions are available as well [see 101, for a recent survey on DFMs]. The next chapter provides a more detailed description of DFMs and their estimation methodologies. However, in most of the applications

imposed on the state and observable equations, and of course in the interpretation of the estimated parameters. In DSGE these restrictions are driven by economic theory and the economic interpretation of the latent factors. On the other hand, in DFMs these restrictions are driven by the underlying statistical assumptions about the data generating process. Notwithstanding these differences, in *Kryshko* [102] a data-rich DSGE model [103] with a standard New Keynesian core is compared empirically to a DFM on a rich panel of U.S. macroeconomic and financial data [104]. The work finds that the space spanned by the statistical factors of the DFM and the space spanned by economic factors of the data-rich DSGE are actually very close.

Recent works have applied Deep Learning to macroeconomic questions. *Cook and Hall* [17] employed a number of neural network architectures, including autoencoders, to forecast the U.S. unemployment rate. *Loermann and Maas* [18] proposed a neural net model to predict the U.S. GDP. *Coulombe* [105] describes a neural network architecture for the estimation of the Philips curve and the output gap. *Hauzenberger et al.* [106] use different dimensionality reduction techniques including autoencoders[4] to construct features for inflation forecasting. *Holopainen and Sarlin* [19] carried out a horse race among different machine learning methods and showed that such models are able to outperform conventional statistical approaches in predicting crisis periods. In *Fernández-Villaverde and Guerrón-Quintana* [82] different methods are discussed for the estimation of DSGE, including machine learning and deep learning ones. Finally, it is worth mentioning that some recent studies have considered the use of alternative data (e.g., text and google data) in DFMs [see 107, 51, for example].

these assumptions are made to facilitate the estimation.

[4]Those are self-supervised deep learning models that will be introduced formally in the methodological background chapter.

2.3 Portfolio Optimisation and Machine Learning

The emergence of Finance as a distinct theoretical and practical discipline with respect to Economics was initiated by Harry Markowitz's article 'Portfolio Selection' in 1952, what is known as the *big bang* of Finance [108]. In his article, Markowitz defines for the first time return and risk. The former being related to the expected value of a distribution, while the latter identified with the variance or the squared deviations around the mean of such distribution. As noted by *Miller* [108], at that time the definition of variance as measure of risk may have appeared counterintuitive to many whose perception of risk coincides with the likelihood of losses, the downside risk. However, in front of a Normal distribution the downside and upside risk are symmetrical, as in any symmetric distribution. With this connection between return-risk and mean-variance, Markowitz introduces mathematical statistics to the study of the portfolio selection problem. Later on William Sharpe builds on Markowitz's work to show that in a world where investors have all mean-variance preferences and form estimates in the same manner and conditioned on the same information set, they will all hold the same portfolio: the market portfolio. Differences in risk aversion among investors will only impact the portion invested in risky assets as opposed to the risk-less bond, but not the composition of the risky assets portfolio. Therefore, this implies that there is just one risk factor, the market, and the distribution of expected rates of return across all risky assets could be expressed as a linear combination of this risk factor, plus a diversifiable idiosyncratic component. This is known as the Capital Asset Pricing Model (CAPM). The introduction of the CAPM from William Sharpe based on the work of Harry Markowitz lent the ground for the empirical and theoretical investigations in Finance. Since then, new risk factors have been discovered, and the number of such factors have been increasing with an out-of-control rate, as documented in a recent census from *Harvey and Liu* [109]. Among the many factors present in the financial economic literature, the commonly

accepted are the betting against beta (BAB) factor from *Frazzini and Pedersen* [35], the market (MKT), small minus big or size (SMB), and high minus low or value (HML) factors from the Fama–French three-factor model [31, 32, 33], and the up minus down or momentum (UMD) factor introduced by *Carhart* [34].

A formulation of the portfolio optimisation problem consists in finding the series of vectors representing asset allocation weights $\{\mathbf{w}_t\}_{t\geq 0}$ that maximise an expected lifetime utility subject to some constraints.[5] This problem involves the estimation of future wealth distribution, which in turns is a function of asset allocation weights and future returns' joint distribution; thus, the task, implicitly or explicitly, involves also a prediction problem. Therefore, one can think of it as a combination of two problems: a forecasting problem and an optimisation problem given these forecasts. However, the problem can also be considered from the viewpoint of a unique reward optimisation problem under the lens of Reinforcement Learning. In particular, one can see the portfolio optimisation problem under the lens of stochastic control and therefore interpret the problem as a Markov Decision Process (MDP). Namely, one can collect in a stochastic vector $\mathbf{x}_t$ the states relevant for a given transition probability function that conditioned on those and the actions taken at time t will provide the next state $\mathbf{x}_{t+1}$ and the reward achieved in transitioning from the current state to the new state, given the action taken. Within the portfolio optimisation context the action space could be interpreted as the feasible allocation weights set. Those actions are expressed as the output of a function of the current states which is called the policy function. On the other side, the state vector $\mathbf{x}_t$ should contain all features necessarily to specify the state transition probabilities and rewards. Although this stochastic control problem could be approached with standard tools from Dynamic Programming, the dimensionality of the portfolio allocation problem makes the computational cost of these methods impossible to sustain. Therefore, approximate methods are more appropriate and the representation of the portfolio optimisation task as a stochastic control problem

[5]Within this thesis we focus on self-financing portfolios, thus assuming income and consumption are zeros across all periods [110].

unleashes the power of approximate methods from the Reinforcement Learning (RL) literature [see 111, 112, 113, for example]. In practice these approaches are sensible to algorithmic design, the features representing the states, the action sets definition and the design of the reward function that aims to represent and/or approximate some or all aspects of the true (lifetime) utility function of the investor. Within the field of RL, when deep neural networks are used to parametrise some or all parts of the process the overall approach takes the name of Deep Reinforcement Learning (DRL).

Hu and Lin [114] outline and address some of these issues in the context of DRL for portfolio management. Additionally, financial data are not identically and independently distributed, and this could prohibit the use of randomised cross validation methods for hyper-parameter tuning.[6] Moreover, in its original form the problem is not first order Markov, and it needs to be converted from an M^{th} order Markov.[7] Also, in financial applications a full state space representation is seldom if ever possible to obtain, given the unknown high dimensional features of the economic and financial world. Therefore, more than (convertible to) fully observable MDP we will be faced with a problem that can be at most converted to a partially observable MDP (POMDP). This makes standard RL tools prone to possible failures if applied blindly. The authors consider also the problem of the reward design and discuss the use of different metrics to approximate a general utility function including the Sharpe ratio. They consider different DRL approaches including policy gradient methods and actor-critic methods. The former optimises directly a parametrised policy function with respect to the designed reward. In the latter the policy function is used by the actor to output an action which is then evaluated by a critic

[6]These models have a number of hyperparameters whose values are selected on the so called validation set(s) generally produced via a method called cross-validation. The problem of cross-validation in time series is discussed also in *Bergmeir et al.* [115]; while, *Bailey et al.* [116] discuss the related issue of backtesting in Finance.

[7]A stochastic process is said to satisfy the first order Markov condition (from Andrey Andreyevich Markov) if at any time t the conditional probability of an arbitrary future event given the entire past of the process is equal to the conditional probability given only the present state values [see 117, for example].

network.

According to *Almahdi and Yand* [118], the first applications of dynamic programming methods to the asset allocation problem are discussed in *Bertsekas* [119], while early applications of RL can be found in *Moody et al.* [120, 121]. More recently, *Huang* [122] proposes to solve the problem with a state of the art DRL algorithm: Deep Recurrent Q-Network (DRQN). In an application to the exchange rate market with data at the 15 minutes frequency, they define the state space as a 198 dimensional vector which includes time features, market features including past closing and tick volumes of each currency, and position features representing the current allocation of the investor. The action space allows for changes in these allocations. Rewards are defined as portfolio log-returns net of transaction costs, which are the product of a constant spread and the absolute change in allocations between two consecutive periods. The DRQN is parametrised with LSTM layers and the authors show the potential of this framework.

Chen et al. [123] propose agent-based reinforcement learning for trading via cloning trading strategies from professionals. *Chakraborty* [124] evaluate the profitability of various Reinforcement Learning (RL) algorithms, including Q-learning and SARSA. They show that such approaches can beat the buy and hold strategy. *Jeong and Kim* [125] propose a framework that combines deep neural networks, RL and Transfer Learning (TL) to build a financial trading system for profit maximisation. The authors identify in TL a way to address the small data problem in Finance. *Ritter* [126] shows in a simulated market the potential of reinforcement learning solutions and Q-learning in particular.

Song et al. [127] analyse portfolio allocation from the perspective of a ranking problem to be then converted to a long-short strategy. They adopt two ranking algorithms: ListNet and RankNet. They apply the two algorithms using 10 years of market and news sentiment data. In a backtesting assessment from 2006 to 2014, they show that their strategies produce risk-adjusted returns superior to the S&P500 index and a measure of the hedge fund industry average

performance.

Almahdi and Yang [118] extend the existing work in RL to build a portfolio optimisation framework that accounts for transaction costs and features adaptive stop-loss to consistently outperform the hedge fund index benchmark. Moreover, they suggest a reassessment of the previously proposed RL objective functions, including those based on the Sharpe ratio. They propose to use the Calmar ratio, which differs from the Sharpe ratio with respect to the denominator adopted to adjust returns by risk. In particular, the Sharpe ratio adjust returns by taking a ratio of the expected returns to the standard deviation; while, the Calmar ratio uses the expected maximum drawdown as denominator. They show this latter specification to deliver superior return performances.

Within the context of Deep Reinforcement Learning, *Jiang et al.* [128] introduce a framework that aims to address the portfolio allocation issue using three types of neural networks: CNN, a basic RNN, and LSTM. *Liang et al.* [129] analyse different deep reinforcement learning algorithms, including Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO) and Policy Gradient (PG). In an empirical evaluation on the Chinese stock market, they found that the standard approaches that are useful in game playing and robot control contests do not apply in this financial problem. In particular, they show that PG is more desirable in financial market than DDPG and PPO, although the latter being more advanced. Finally, they propose adversarial training methods and a PG algorithm to outperform the uniform constant rebalanced portfolio. *Heaton et al.* [42] support the idea that deep learning tools can detect and possibly exploit interactions in the data that do not seem to be captured by existing financial economic theory. They show that a portfolio weighted using deep learning outperformed the IBB index. *Deng et al.* [130] use a DRL approach by combining recurrent networks with direct policy learning and show successful applications on the S&P 500 and on commodity future contracts.

Wang et al. [131] propose a combination of LSTM and mean-variance in

an application to the components of the UK Stock Exchange 100 Index with data from 1994 to 2019. The LSTM selects the assets with higher expected returns. They compare against other ML tools, such as support vector machine, random forest, deep neural networks, and autoregressive integrated moving average models. They show LSTM can outperform all the others. On the selected assets a mean-variance optimisation is carried out, and the authors find this LSTM mean-variance strategy to outperform five baselines across all the performance measures used, including cumulative returns and Sharpe ratio.

Different approaches are adopted in *De Prado* [41] and *Raffinot* [132]. Here the authors employ hierarchical clustering algorithms to construct diversified portfolios. Namely, a variation of risk parity [41] and a variation of the equal risk contribution [132], both aiming to take advantage of the possible hierarchical correlation structure among assets, are introduced. Finally, other Machine Learning approaches to the problem include also the application of evolutionary algorithms [see 133, 134, 135, for example] and quantum annealing techniques [135] in order to take into consideration additional constraints that are faced in the practical implementation process.

2.4 Explainability in Machine Learning for Economics and Finance

Notwithstanding eXplainable AI (XAI)[29] has experienced growing interest in recent years [30], its applications to economic and financial problems has seen slower development. The following are some few exceptions in this context.

In *Nakagawa et al.* [136] an explainable Deep learning framework is proposed. The authors construct Deep Factor models by employing fully connected multilayer perceptrons (MLP) to predict the next month's stock returns given present and past historical returns, together with 17 factor exposures including risk, quality, momentum, value and size. They show these deep learning architectures to be superior to a linear model, a supported vector regression (SVR) and a random forest (RF) both on accuracy and profitability

on the constituents of the TOPIX, a stock market index for the Tokyo Stock Exchange. Finally, they analyse factor exposures by means of Layer-Wise Relevance Propagation (LRP) [137]. This is a feature attribution method aiming at additively decomposing a given prediction with respect to its drivers. As such, their paper can be classified in the explainable space, as the model on its own is not interpretable by default and necessitates an ex-post interpretable representation, in this case by means of a feature attribution method. An example of an interpretable model in the context of feature attribution is the linear regression, where the additive decomposition of the contribution is provided by the model itself and no ex-post external decomposition is required to the purpose.

Nakagawa et al. [138] extend the previous work by including both LSTM and interpretable versions of both LSTM and fully connected multilayer perceptrons by means of LRP. Namely, they express asset returns as a linear function of LRP relevance scores; thus, proposing an interpretable nonlinear and time varying multi-factor model. In an empirical analysis on the TOPIX500 constituents they show the proposed LSTM+LRP to beat a standard LSTM, an MLP, an MLP+LRP, a linear model, an SVR and a RF in terms of higher Sharpe Ratio, higher returns and lower volatility. This work stays in contrast with the previous one. Indeed, here *Nakagawa et al.* [138] use the forecasts of an interpretable model, being it a linear combination of feature attribution scores computed via LRP. This differs from the previous work [136], where the authors use a model for prediction which is then explained via LRP for determining factor attributions. These two works highlight the difference between direct interpretable models, and models that necessitate an explainable tool to uncover some aspects of the predictions made.

In *Bracke et al.* [139] the authors extend the work of *Datta et al.* [140] and define clustering based Shapley attributions to compute groups of explanations in the context of a mortgage default prediction problem. Shapley method is a feature attribution method that similarly to LRP provides a local additive

decomposition of the model predictions with respect to its input variables. *Bussmann et al.* [141] show how to combine Shapley values with correlation networks for explanability in a credit risk management context to explain the predictions of an Extreme Gradient Boosting (XGBoost).

In *Hoepner et al.* [142] the concepts of significance, relevance and explainability are analysed both under statistical and economic lens. The authors advocate for more research towards fully transparent and entirely replicable algorithms, where statistical tests can be carried out and the mechanisms of the algorithms can be entirely understood by the human decision maker.

Chapter 3

Methodological Background

In this chapter we cover the common background material on which we build the subsequent chapters. In Section 3.1 we review dynamic factor models, in Section 3.2 deep learning models, whereas, Section 3.3 is dedicated to portfolio optimisation theory and approaches. We conclude this chapter with Section 3.4 about explainability in Machine Learning with a focus on Deep Learning.

3.1 Overview of Dynamic Factor Models

Following *Doz and Fuleky* [101] Dynamic Factor Models (DFMs) can be broadly defined as parsimonious representations of relationships among time series variables. The method was firstly introduced in psychology by *Spearman* [143], who used a single factor or unobservable variable to describe cognitive abilities. It was then extended by *Geweke* [83] to capture co-movements in economic time series data. The actual idea that co-movements in such series can be linked to business cycle was originally described by *Burns and Mitchell* [144].[1] Early applications of DFMs to such data can be found in *Sargent and Sims* [84] and *Stock and Watson* [145, 38, 146], while among the first applications to financial data there is *Chamberlain* [147] and *Chamberlain and Rothschild* [148]. Very good surveys have been written on DFMs, and while referring to those for further details [see 149, 150, 151, 101, for example], in what follows we briefly

[1] In particular, the authors describe cycles as follows: "a cycle consists of expansions occurring at about the same time in many economic activities, followed by similarly general recessions, contractions, and revivals which merge into the expansion phase of the next cycle; this sequence of changes is recurrent but not periodic."

describe the different estimation methods to then focus on one specific instance of them which is the starting point of one of the contribution of this thesis.

Before discussing estimation methods, we highlight some differences between various factor models and some of the common assumptions. Let $\{\mathbf{y}_t\}_{t\geq 0}$ be a $n-$dimensional vector of time series assumed to be weakly-stationary, with zero mean and unit variance.[2] DFM assumes that the following decomposition holds

$$\mathbf{y}_t = \boldsymbol{\zeta}_t + \boldsymbol{\varepsilon}_t \tag{3.1}$$

The process $\{\boldsymbol{\zeta}_t\}_{t\geq 0}$ is called the *common component*. It is a function of the factors which are generally assumed to be stationary. The process $\{\boldsymbol{\varepsilon}_t\}_{t\geq 0}$ is a $n-$dimensional vector collecting the so called *idiosyncratic processes*. A factor model has a static and a dynamic representation. In dynamic factor models we have $\boldsymbol{\zeta}_t = \boldsymbol{\Theta}(L)\boldsymbol{u}_t$, where $\boldsymbol{\Theta}(L)$ is the factor loading polynomial matrix and $\boldsymbol{u}_t$ is an $d-$dimensional vector of common dynamic shocks at time t with $d << n$ [see 88, 87, 89, 90, 91, for example]. In the static representation of the DFM [85, 152] the common component is assumed to satisfy the following equation

$$\boldsymbol{\zeta}_t = \boldsymbol{\theta}\boldsymbol{f}_t \tag{3.2}$$

with

$$\boldsymbol{f}_t = \boldsymbol{B}(L)\boldsymbol{f}_t + \boldsymbol{u}_t = \boldsymbol{B}_1\boldsymbol{f}_{t-1} + \cdots + \boldsymbol{B}_p\boldsymbol{f}_{t-p} + \boldsymbol{u}_t \tag{3.3}$$

where $\boldsymbol{f}_t$ is the time t, $d-$dimensional vector of common factors with $d << n$.

The common and idiosyncratic components are always assumed to be orthogonal to each other, while another difference made in the DFM literature is among exact and approximate factor models. The exact factor model further assumes that the idiosyncratic processes are also orthogonal to each other, so that all the correlations among the observables are captured by the common factors only. When mild cross-correlation among the idiosyncratic components is allowed, the model is called approximate factor model.

[2] See [37] for an example of a DFM with non-stationary series.

3.1.1 Estimation Methods

Different estimation methods have been proposed in the literature for DFMs, here we summarise some of them while referring to the surveys mentioned before for a more detailed treatment.

The static exact factor model can be estimated via maximum likelihood under the assumption of $\{\boldsymbol{f}_t\}_{t\geq 0}$ and $\{\boldsymbol{\varepsilon}_t\}_{t\geq 0}$ being two orthogonal *i.i.d.* Gaussian processes [101]. The unique identification of the model is achieved via some restrictions. One of which being the fact that the idiosyncratic components are mutually orthogonal processes, thus implying a diagonal covariance matrix for $\boldsymbol{\varepsilon}_t$. A second restriction imposes the factors $\{\boldsymbol{f}_t\}_{t\geq 0}$ to have their covariance matrix equal to the identity matrix, $Var(\boldsymbol{f}_t) = \mathbf{I}_d$, estimates can be then obtained by means of numerical methods [101]. Also a dynamic exact factor model can be estimated by maximum likelihood under the gaussianity assumption. In particular, the gaussianity assumption allows to write factor models as linear gaussian state spaces, where factors are represented as latent states. Therefore, estimates of the values of the latent factors and the gaussian likelihood can be computed using Kalman filter. *Watson and Engle* [153] propose different methods including the EM algorithm. Despite this progress, these early applications had two limitations: one computational and the other conceptual [150]. For the latter, the maximum likelihood estimation requires a full parametric specification, which translates in Gaussian disturbances and mutually independent idiosyncratic processes (exact as opposed to approximate factor model). However, *Doz et al.* [154] show that maximum likelihood estimator from the Gaussian state space formulation, and more in general quasi-maximum likelihood estimator, is a consistent estimator of the space spanned by the factors under weak assumptions on the error distribution and allowing for limited correlation in the idiosyncratic components.

An alternative non-parametric method for the estimation of approximate static factor models is provided by principal component analysis (PCA). Generally, the constraints imposed to recover a unique solution are in the form of

normalising conditions, such as $\frac{1}{T}\sum_{t=1}^{T}\hat{\boldsymbol{f}}_t'\hat{\boldsymbol{f}}_t = \mathbf{I}_d$. More so, since PCA is not scale invariant, the original series are usually centered and standardised before applying PCA. Early application of this method are from *Chamberlain and Rothschild* [148], while *Stock and Watson* [85, 86] and *Bai and Ng* [152] popularise the method to macroeconomics and derive consistency of the estimates. Extensions to deal with heteroskedastic and serially correlated idiosyncratic terms have been analysed [see 155, for example]. Even though PCA has been regarded as a building block for the estimation of dynamic factor models, it needed extensions as alone it does not capture dynamics [101].

Finally, other approaches carry the estimation in the frequency domain, as the method proposed by *Forni et al.* [88, 89] which is based on the dynamic representation of the model and it consists in two steps, involving the estimation of the spectral densities of the common and idiosyncratic parameters based on the sample autocovariance of the observables $\mathbf{y}_t$.

3.1.2 Likelihood Based Approaches

Doz et al. [36] introduce a two-step quasi-maximum likelihood estimation for approximate dynamic factor models of the following form

$$\mathbf{y}_t = \boldsymbol{\theta}_F \boldsymbol{f}_t + \boldsymbol{\varepsilon}_t \ , \tag{3.4}$$

$$\boldsymbol{f}_t = \boldsymbol{B}_1 \boldsymbol{f}_{t-1} + \cdots + \boldsymbol{B}_p \boldsymbol{f}_{t-p} + \boldsymbol{u}_t, \tag{3.5}$$

where they allow the idiosyncratic terms to be autocorrelated. The estimation works as follows

Step 1 Compute the preliminary estimator of $\hat{\boldsymbol{\theta}}_F$ and $\hat{\boldsymbol{f}}_t$ via principal component analysis. The idiosyncratic is then given by $\hat{\boldsymbol{\varepsilon}}_t = \mathbf{y}_t - \hat{\boldsymbol{\theta}}_F \hat{\boldsymbol{f}}_t$ and their variance is obtained from the empirical variance. The dynamics of the common factors are estimated using vector autoregressive.

Step 2 The model is cast in state-space where it is assumed that the variance of the common shocks is the identity matrix $cov(\boldsymbol{u}_t) = \boldsymbol{U} = \boldsymbol{I}_d$, and the

idiosyncratic components have a diagonal matrix. Conditioned on the parameters, the Kalman smoother provides a new estimate of the factors. The idiosyncratic terms here are assumed to be mutually orthogonal white noises.

In *Doz et al.* [154] a quasi-maximum likelihood method for the estimation of large scale approximate dynamic factor model is proposed. The method is derived based on a maximum likelihood estimator for mutually orthogonal *i.i.d.* gaussian idiosyncratic terms and gaussian VAR for the common components. The log-likelihood of this linear gaussian state space model is obtained via the Kalman filter and an EM algorithm is implemented to get the maximum likelihood estimator, thus allowing successive applications of the two-step approach. The author prove the mean square consistency of the estimators under standard assumptions. Further, they show that the validity of the results is maintained even under non-gaussianity or non-*i.i.d.* of the idiosyncratic terms, as long as the cross-correlation is limited.

A problem with economic data is the presence of missing observations and *Banbura and Modugno* [39] adapted the EM algorithm to a general pattern of missing data by using selection matrix when carrying out the maximisation step. In the E-step the algorithm behaves as if the data were complete, while the missing data are replaced by the best linear fit given the information set. The authors also extend the approach to the case where AR processes are assumed for idiosyncratic components. Moreover, they show how to compute a statistical decomposition to measure the contribution to the forecast movements from the new unexpected information, the so called *news*. We will use this framework as our benchmark and starting point [93, 39]

$$\mathbf{y}_t = \boldsymbol{\theta}_F \boldsymbol{f}_t + \boldsymbol{\varepsilon}_t \ , \tag{3.6}$$

$$\boldsymbol{f}_t = \boldsymbol{B}_1 \boldsymbol{f}_{t-1} + \cdots + \boldsymbol{B}_p \boldsymbol{f}_{t-p} + \boldsymbol{u}_t, \qquad \boldsymbol{u}_t \overset{iid}{\sim} \mathcal{N}(0, \boldsymbol{U}), \tag{3.7}$$

$$\boldsymbol{\varepsilon}_t = \boldsymbol{\Phi}_1 \boldsymbol{\varepsilon}_{t-1} + \cdots + \boldsymbol{\Phi}_d \boldsymbol{\varepsilon}_{t-d} + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \overset{iid}{\sim} \mathcal{N}(0, \boldsymbol{Q}), \tag{3.8}$$

where $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_p$ are the $r \times r$ matrices of autoregressive coefficients for the factors and $\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_d$ are the $n \times n$ diagonal matrices of autoregressive coefficients for the idiosyncratic component (i.e. $\boldsymbol{\Phi}_1 = diag(\phi_1, \ldots, \phi_n)$). Specifically, the authors [93, 39] assume a VAR process of order two ($p = 2$) for factors, and of order one ($d = 1$) for the idiosyncratic components.[3]

The likelihood of the linear Gaussian state space model described above is invariant to any invertible linear transformation of the factors [see 93, for example]. These means that the parameters are not identifiable from the data as any invertible linear transformation of them is observationally equivalent. Therefore, the EM will converge to one of these transformations [156]. In forecasting applications this identifiability issue is not a problem, as one is not interested in the factor themselves but only in the space spanned by those [93]. Nevertheless, additional restrictions can be easily imposed on the parameter space to achieve their identifiability [see 157, 158, 93, for example].

3.2 Overview of Deep Learning Models

The history of Deep Learning (DL) can be traced back to stochastic gradient descent in 1952, which is employed for an optimisation problem. At that time the limit of computer hardware prevented its potential applications. Today, the developments of graphics processing units (GPUs), dataset storage and processing and distributed systems allows effective DL applications to different fields such as medicine, neuroscience, physics and astronomy, finance and banking, and operations management [2].

In order to review deep learning models relevant to this PhD thesis, we begin with the feed-forward multilayer perceptron; then we briefly review other types of networks.[4] We then present other two building blocks of this PhD thesis: autoencoders and policy-based Deep Reinforcement Learning.

[3]The estimation method has been shown to be robust to the zero cross-correlation assumption at all leads and lags of the idiosyncratic components when mildly violated in small sample [154, 37].

[4]We refer to *Goodfellow et al.* [159] for further details.

3.2.1 Multilayer Perceptrons

The simplest Deep Learning model is the so called *multilayer perceptron (MLP)*, formally designated as making the following point prediction

$$\widehat{\mathbf{y}}_t = g_{\boldsymbol{\theta}_L}(\dots g_{\boldsymbol{\theta}_1}(\mathbf{x}_t)) \tag{3.9}$$

where $\hat{\mathbf{y}}_t$ is the output vector[5], $\mathbf{x}_t$ is the input vector, and $G_{\boldsymbol{\theta}}(\cdot) = g_{\boldsymbol{\theta}_L}(\dots g_{\boldsymbol{\theta}_1}(\cdot))$ is a composite operator formed by the composition of L functions and their related parameters (one for each layer $l = 1, \dots, L$ after the input). Those functions are called *link functions* or *activation functions*. The activation functions can differ between layers and can be both linear and nonlinear. Each activation function produces an *activation output* which is the product of an element wise (usually monotone) transformation ($g_l(\cdot)$) applied on an affine mapping (matrix $\boldsymbol{\theta}_l$) of the neurons in the lower layer.

Namely, if we assume $h_t^{m_l}$ is the neuron m_l in layer l at observation t, then

$$h_t^{m_l} = g_l(\boldsymbol{\theta}_{m_l} \cdot ([1]||\boldsymbol{h}_t^{l-1})) \tag{3.10}$$

$\forall m_l = 1, \dots, k_l$, where k_l is the number of neurons in layer l. θ_{m_l} is the m^{th} row of matrix $\boldsymbol{\theta}_l$. Therefore, the activation output of each neuron, $h_t^{m_l}$, depends on three different elements: the activation function $g_l(\cdot)$, the vector of weights and biases related to that neuron, $\boldsymbol{\theta}_{m_l}$, and the output vector of the previous layer, $\boldsymbol{h}_t^{l-1}$. By hierarchically repeating this operation for each layer $l = 1, \dots, L$, we obtain the MLP output: $G_{\boldsymbol{\theta}}(\mathbf{x}_t)$.

The aim of the model is to infer the correct state of the target variable (i.e., produce an accurate estimate), which is achieved by minimising a loss function with respect to the tensor of parameters $\boldsymbol{\theta}$:

$$\mathcal{L}(\mathbf{y}_t, \widehat{\mathbf{y}}_t) = \mathcal{L}(\mathbf{y}_t, G_{\boldsymbol{\theta}}(\mathbf{x}_t)). \tag{3.11}$$

[5]Or a scalar in case of single target or binary class.

The most commonly used loss functions are the squared loss for regression, it being proportional to the likelihood of the model when assuming conditional Gaussianity, and cross entropy for classification. The minimisation of the loss function with respect to θ is performed via *back-propagation* [160, 161], thus requiring the loss function to be *almost everywhere differentiable*. Although different optimisation algorithms can be used, the first ones were based on gradient descent [162] which uses the entire training dataset, called *batch*, to update the gradient of each parameter. Afterwards, to speed up the optimisation procedure, stochastic gradient descent (SGD) algorithms [163] that update the gradient of each parameter using only randomly selected subsamples of the training dataset have been applied. These subsamples are called *minibatches* and are an equal partition of the original training dataset. Thus, for each iteration SGD has a computational cost which is independent with respect to the sample size; that is, if we assume a mini batch of one observation, SGD has computational cost of $O(1)$, while GD's computational cost is $O(n)$ where n is the number of observations. SGD sometimes can remain slow or get stuck when the optimisation reaches a particular point of the loss function, e.g., flat regions or saddle points. Therefore, the method of *momentum* [164] is designed to possibly overcome these issues and accelerate the learning. Momentum algorithms add an exponentially decay moving average of past gradients by introducing a velocity element. In the literature, different momentum algorithms have been proposed (AdaGrad [165] and RMSProp and its variation [166]), but ADAM [167] is the most popular because it combines the advantages of the others. Indeed, it can suit multiple different datasets reasonably well with little or no tuning of its hyperparameters [168]. ADAM updates the gradient using bias correction of its first moment, through the weighted moving average, and its raw second moment, through the uncentered variance. By doing so, it adapts the learning rate in each dimension directly proportional with respect to the first moment and inversely proportional with respect to the second moment. All the optimisation algorithms tend to analyse the training dataset multiple

times in order to reach a better estimation of the parameters. A run of the algorithm over the entire dataset is called an *epoch*.

3.2.2 Other Types of Networks

In addition to the fully connected MLP, other common types of networks are *convolutional neural networks* (CNNs) [169, 170] and *recurrent neural networks* (RNNs) [160].

Differently from a fully connected MLP, CNNs enjoy parameter sharing and local connections on input data with known grid-like topology. Some of the successful early applications of such models were on handwritten characters [171, 172, 173]. Applications of such models do not need to be on image data only. Indeed, in *Simard and Le Cun* [174] convolutional networks are used for time signals. Variants of such models are known as *unshared convolution* and *tiled convolution*. In unshared convolutions there is local connection without parameter sharing [159]. While tiled convolutions [175, 176] offer different convolution operators on adjacent inputs that are shared on the remaining ones.

Whereas recurrent neural network (RNNs) are a family of networks for processing sequential data [159]. The idea behind this type of modelling is to share parameters across time. They express the current observation as a function of a hidden state that has a recurrent structure. Therefore, information is stored from the initial state, until the current one. However, the propagation of the model's gradients over many time steps to learn long term dependencies could easily make those gradients vanish or explode. To solve this problem, *long short-term memory* models (LSTM) [177, 178] and their compressed versions - *gated recurrent units* (GRUs) [179, 180, 181, 182] - were introduced. These models use self-loops to make the gradient flow, partially or totally, to the distant past.

As in this work we will make use of LSTM layers as well, we now provide a formal definition of an LSTM layer. Namely, its update equations can be

expressed as follows

$$\begin{aligned}
\boldsymbol{f}_t &= sigmoid(\boldsymbol{W}_{f,x}\mathbf{x}_t + \boldsymbol{W}_{f,h}\boldsymbol{h}_{t-1} + \mathbf{b}_f) \\
\tilde{\boldsymbol{c}}_t &= sigmoid(\boldsymbol{W}_{\tilde{c},x}\mathbf{x}_t + \boldsymbol{W}_{\tilde{c},h}\boldsymbol{h}_{t-1} + \mathbf{b}_{\tilde{c}}) \\
\boldsymbol{i}_t &= sigmoid(\boldsymbol{W}_{i,x}\mathbf{x}_t + \boldsymbol{W}_{i,h}\boldsymbol{h}_{t-1} + \mathbf{b}_i) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tilde{\boldsymbol{c}}_t \\
\boldsymbol{o}_t &= sigmoid(\boldsymbol{W}_{o,x}\mathbf{x}_t + \boldsymbol{W}_{o,h}\boldsymbol{h}_{t-1} + \mathbf{b}_o) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \tanh(\boldsymbol{c}_t)
\end{aligned} \tag{3.12}$$

where the $\boldsymbol{W}$'s represent matrices of weight parameters, and $\mathbf{b}$ are vector of biases, while *sigmoid* represent the sigmoid function and tanh the hyperbolic tangent function. $\mathbf{x}_t$ and $\boldsymbol{h}_t$ represent the input vector and the hidden state respectively. $\boldsymbol{f}_t, \boldsymbol{i}_t, \boldsymbol{o}_t$ are the gates called forget, input and output. Finally $\boldsymbol{c}_t$ is the so called cell state and $\circ$ is the Hadamard product (element-wise product). It is common to add on top of this layer a SLP (linear for regression, sigmoid or softmax for classification) or a MLP to make a prediction of the following form $\widehat{\mathbf{y}}_t = G_{\boldsymbol{\theta}}(\boldsymbol{h}_t)$.

3.2.3 Autoencoders

Autoencoders (AEs) belong to the DL family of models and have been introduced for applications involving dimensionality reduction [161, 183, 184, 185]. Autoencoders solve the parametric problem of finding a mapping (or learning a representation) of the form $\widehat{\mathbf{y}}_t = F_{\boldsymbol{\theta}_F}(G_{\boldsymbol{\theta}_G}(\mathbf{y}_t))$ by minimising a loss function of choice with respect to the parameters $\boldsymbol{\theta}_F$ and $\boldsymbol{\theta}_G$

$$\mathcal{L}(\mathbf{y}_t, \widehat{\mathbf{y}}_t) = \mathcal{L}(\mathbf{y}_t, F_{\boldsymbol{\theta}_F}(G_{\boldsymbol{\theta}_G}(\mathbf{y}_t))) \ . \tag{3.13}$$

Here $G_{\boldsymbol{\theta}_G}(\cdot)$ is called the *encoder* network, as it maps the original inputs to a lower dimensional code that synthesises relevant information, whilst $F_{\boldsymbol{\theta}_F}(\cdot)$ is the *decoder* network that maps the low dimension code back to its original higher dimensional space. The *Principal Components Analysis*

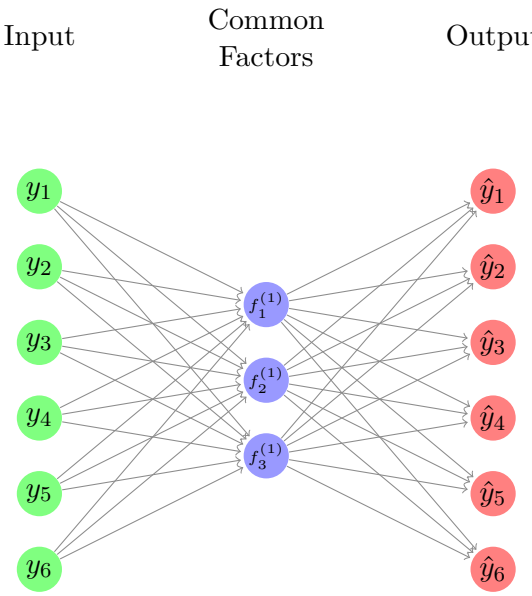


Figure 3.1: Principal component analysis (PCA) as an autoencoder.

(PCA) can be seen as the autoencoder minimising the squared loss function (i.e.: $\mathcal{L}(\mathbf{y}_t, \hat{\mathbf{y}}_t) = ||\mathbf{y}_t - \hat{\mathbf{y}}_t||^2$) and assuming that both the encoder and the decoder are linear networks without bias terms, i.e. $\boldsymbol{f}_t = G_{\boldsymbol{\theta}_G}(\mathbf{y}_t) = \boldsymbol{\theta}_G \cdot \mathbf{y}_t$ and $\hat{\mathbf{y}}_t = F_{\boldsymbol{\theta}_F}(\boldsymbol{f}_t) = \boldsymbol{\theta}_F \cdot \boldsymbol{f}_t$. Figure 3.1 shows a representation of PCA as an autoencoder. Indeed, as discussed in *Baldi and Hornik* [186], affine decoder and encoder without any nonlinearity and squared error loss will recover the same subspace of PCA. Moreover, when nonlinearity is added into the encoding network, PCA appears as one of the many possible representations [183, 187].

Since in principle the two functions that compress and decompress the data can be any kind of function, finding the correct functional form which is capturing the data generating process of interest can be a daunting problem. Autoencoders provide a practical implementation of this problem by expressing the composition of two functions as a chain of two networks: the first chain operates the encoding, while the second produces the decoded output (see a graphical representation of a symmetric fully connected autoencoder in Figure 3.2).

Although the functional form adopted for the activation functions may seem arbitrary, a network with such a structure can nevertheless approximate

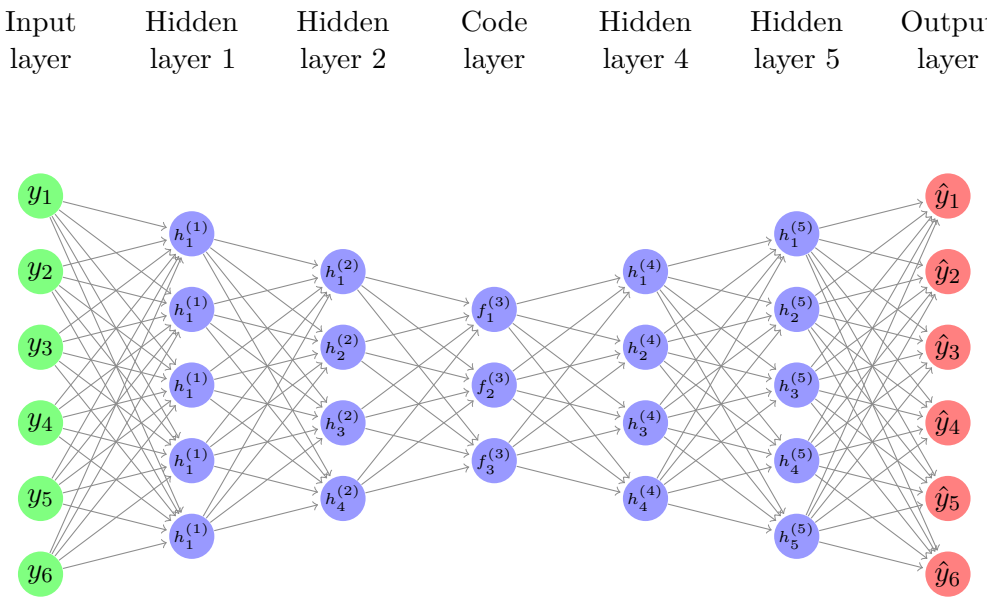


Figure 3.2: A symmetric autoencoder with six observables and three neurons in the code layer (biases are not included in the graph). The first two hidden layers operate the encoding, while the last two hidden layers decode into the output.

any nonlinear continuous function. In fact, the Universal Approximation Theorem (UAT) – a key result in the neural net literature – states that a feed-forward network, even with a single hidden layer containing a finite number of neurons, can approximate continuous functions on compact subsets of $\mathbb{R}^n$ under mild assumptions on the activation function. However, the UAT does not guarantee that the algorithm adopted to estimate the network can learn the correct parameters [188, 189, 190, 191].

An autoencoder is said to be *symmetric* when the number of hidden layers in the encoding and in the decoding networks are the same; otherwise it is asymmetric. *Asymmetric* autoencoders usually have several layers of encoding but only a single layer of decoding (i.e. the function $G_{\theta_G}(\cdot) = g_{\theta_g}(\cdot)$). They were introduced by *Majumdar and Tripathi* [192] and have been found to be more accurate compared to traditional symmetric autoencoders for classification accuracy, and also able to yield slightly better results on compression problems (over the following datasets: MNIST, CIFAR-10, SVHN and CIFAR-100). Furthermore, the authors [192] argue that the asymmetric

structure helps to reduce the number of parameters to estimate and hence lessen the potential extent of overfitting. The graph in Figure 3.3 shows a fully connected asymmetric autoencoder.

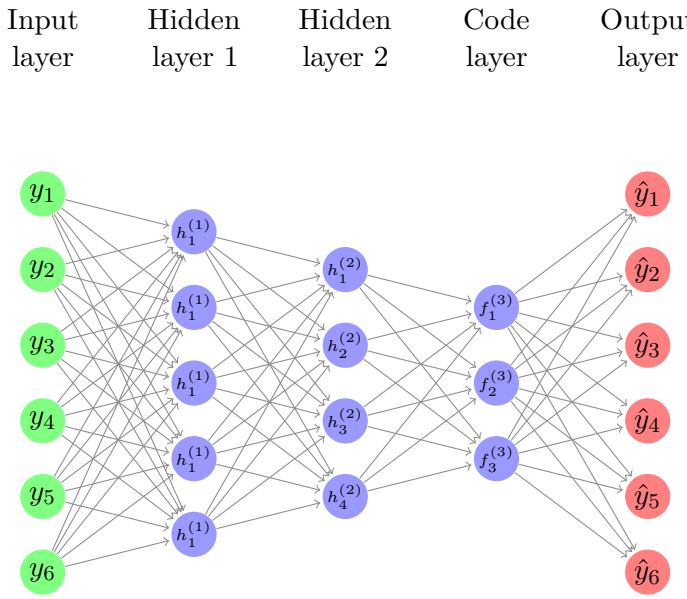


Figure 3.3: Asymmetric Autoencoder with six observables and three neurons in the code layer (biases are not included in the graph).

Denoising Autoencoders (DAEs) [6] are similar to classic autoencoders, but they appear with a more general estimation method. Indeed, DAEs [193] can be seen as AEs with noise injection on the inputs, hence the loss function becomes:

$$\mathcal{L}_{DAE}(\mathbf{y}_t, \hat{\mathbf{y}}_t) = \mathcal{L}(\mathbf{y}_t, F_{\boldsymbol{\theta}_F}(G_{\boldsymbol{\theta}_G}(\tilde{\mathbf{y}}_t))) \tag{3.14}$$

where $\tilde{\mathbf{y}}_t \sim \mathcal{C}(\tilde{\mathbf{y}}_t|\mathbf{y}_t)$ and $\mathcal{C}$ is a given corruption process that maps the real data, $\mathbf{y}_t$ to $\tilde{\mathbf{y}}_t$ by using a conditional distribution. The parameters of the denoising autoencoders ($\boldsymbol{\theta}_F$ and $\boldsymbol{\theta}_G$) can be optimised with standard back-propagation. Denoising has multiple benefits, first of all, as shown by *Vincent et al.* [193] and *Bengio et al.* [194], it forces the model to learn the real data distribution and not only the distribution specific to the sample used; moreover, neural nets are not very robust to noise [195], and noise injection is one way to improve

[6] They can be standard or asymmetric.

robustness. Additionally, it is a useful data augmentation mechanism [196].

Another type of autoencoder is the *Variational Autoencoder (VAE)* introduced by *Kingma and Welling* [197]. This approach estimates the distribution of the latent states by optimising the *evidence lower bound*. *Higgins et al.* [198] have extended the concept to include a hyperparameter to control divergence from the prior distribution, yielding to the β-VAE. The loss function for such an autencoder is expressed as

$$\mathcal{L}_{\beta\text{-VAE}}(\mathbf{y}_t, \hat{\mathbf{y}}_t) = -\mathbb{E}_{\boldsymbol{f}_t \sim q(\boldsymbol{f}_t|\mathbf{y}_t)} log\, p_{decoder}(\mathbf{y}_t|\boldsymbol{f}_t) + \beta D_{KL}(q(\boldsymbol{f}_t|\mathbf{y}_t)||p(\boldsymbol{f}_t)). \tag{3.15}$$

The first term of the loss represents the reconstruction log-likehood. It is related to the probability of observing the data conditioned to the latent states which are distributed according to the encoding network $q(\boldsymbol{f}_t|\mathbf{y}_t)$. The second part of the loss relates to the Kullback-Leibler divergence between the encoding distribution network and the prior on the latent states, $p(\boldsymbol{f}_t)$. If β is set to 1, then we have the standard VAE. Minimising the loss in equation (3.15) coincides with maximising the *evidence lower bound* which is the lower bound of the unconditional likelihood, as it misses the Kullback-Leibler divergence between the encoding network distribution and the true conditional distribution of the latent states given the observable. It is generally argued that VAEs make an approximation of the loss which is arguably small, given high-capacity models, so, they can offer computational advantages with respect to other methods (e.g., Markov Chain Monte Carlo) [199].

3.2.4 Dynamics in Autoencoders

In *Gregor et al.* [200], *Deep AutoRegressive networks (DARN)* are proposed to learn direct probabilistic models for binary latent variables. In their work, hidden layers are equipped with autoregressive connections, thereby allowing for dynamics in an autoencoder setting. Successively, *Temporal Difference Variation Autoencoder (TD-VAE)* was introduced by *Gregor et al.* [201] to model dynamics in autoencoders via (LSTM) connections between belief distributions

at two distant time steps. Those distributions were estimated using VAEs and their linkage is specified via transition distributions and smoothing. But arguably the most relevant to our approach presented in Chapter 4 are *Krishnan et al.* [202], *Fraccaro et al.* [203] and *Rangapuram et al.* [204]. In particular, *Krishnan et al.* [202] adopt multilayer perceptrons (MLPs) to estimate the mean and covariance matrix of a state space with Gaussian transition dynamics. In *Fraccaro et al.* [203] a *Kalman Variational Autoencoder (K-VAE)* is introduced to estimate (locally) linear Gaussian state space models. Similar in spirit to *Lee et al.* [205], where *Structured Variational Autoencoders (SVAEs)* are used to provide conjugate graphical models, in *K-VAE* the authors propose a VAE to disentangle the observations and the latent dynamics. Furthermore, an RNN is used to estimate a time varying linear combination of K linear Gaussian state space models.

3.2.5 Policy-Based Deep Reinforcement Learning

In this section, after giving an introduction to Markow Decision Processes (MDPs) following *Szepesvári* [111], we discuss the REINFORCE algorithm of Ronald J. Williams and its extensions following *Sewak* [113].[7]

We define a countable Markow Decision Process (MDPs) with the triplet $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P}_0)$, where $\mathcal{X}$ is the set collecting the states, $\mathcal{A}$ is the set of actions, while $\mathcal{P}_0$ is the transition probability kernel that for every state-action pair $(\mathbf{x}, \mathbf{a}) \in \mathcal{X} \times \mathcal{A}$ assigns a probability measures over $\mathcal{X} \times \mathbb{R}$. In particular, given a set $\mathcal{U} \subset \mathcal{X} \times \mathbb{R}$, $\mathcal{P}_0(\mathcal{U}|\mathbf{x}, \mathbf{a})$ represent the probability that next state and its associated reward are in set $\mathcal{U}$, conditioned on taking action $\mathbf{a}$ in the current state $\mathbf{x}$. It is then possible to write the state transition probability kernel $\mathcal{P}(\mathbf{x}, \mathbf{a}, \mathbf{y}) = \mathcal{P}_0(\{\mathbf{y}\} \times \mathbb{R}|\mathbf{x}, \mathbf{a})$ which tells us the probability of transiting from state $\mathbf{x}$ to state $\mathbf{y}$ when taking action $\mathbf{a}$. We can then express the expected immediate reward when doing this transition as $r(\mathbf{x}, \mathbf{a}) = \mathrm{E}[R_{(\mathbf{x},\mathbf{a})}]$. MDPs come useful to describe sequential decision making problems. In particular one

[7] We do not cover other Reinforcement Learning algorithms that can be applied to solve MDPs, as those will not be used in this work. We refer the interested readers to standard textbooks on the topic [see 111, 112, 113, for example].

could write $\mathcal{P}(\mathbf{x}, \mathbf{a}, \mathbf{y}) = \mathrm{P}(\mathbf{x}_{t+1} = \mathbf{y} | \mathbf{x}_t = \mathbf{x}, \mathbf{a}_t = \mathbf{a})$, and express the expected reward from this transitioning as $r(\mathbf{x}_t, \mathbf{a}_t) = \mathrm{E}[r_{t+1} | \mathbf{x}_t, \mathbf{a}_t]$. Additionally, after having observed a given behaviour, one could express the return underlying this behaviour as the total discounted sum of the rewards incurred: $\mathcal{R} = \sum_{t=0}^{T} \gamma^t r_{t+1}$. When T is finite we speak about finite horizon MDP, if $T = \infty$ we are in front of an infinity horizon MDP. More so, when $\gamma = 1$ the problem is called undiscounted, while for $0 < \gamma < 1$ the problem is called discounted. The objective of the decision maker is to find a behaviour that maximises the expected return. Behaviours are expressed in terms of a policy function that given the current states provides the action to take $\pi : \mathcal{X} \to \mathcal{A}$. When this policy is deterministic, then $\mathbf{a}_t = \pi(\mathbf{x}_t)$, when it is stochastic then $\mathbf{a}_t \sim \pi(\cdot | \mathbf{x}_t)$. The set of all stationary policy that when put in place deliver to a time homogeneous Markov chain $\{\mathbf{x}_t\}_{t \geq 0}$ is generally defined with Π_{stat}. We can then parametrise a policy π with parameters $\boldsymbol{\theta}$ and define its value as

$$J_{(\boldsymbol{\theta})} = \mathrm{E} \sum_{t \geq 0} [\gamma^t r_t^{(\tau)} | \pi_{\boldsymbol{\theta}}], \tag{3.16}$$

where $\tau = [(\mathbf{x}_0, \mathbf{a}_0, r_1), \ldots, (\mathbf{x}_t, \mathbf{a}_t, r_{t+1}), \ldots]$ is a trajectory influenced by state-transition probabilities under the given policy $\pi_{\boldsymbol{\theta}}$. Hence, the most optimal parameter vector $\boldsymbol{\theta}^*$ that maximises this reward can be expressed as

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} J_{(\boldsymbol{\theta})}. \tag{3.17}$$

One can search for this by computing the following gradient

$$\Delta_{(\boldsymbol{\theta})} J_{(\boldsymbol{\theta})} = \frac{\partial J_{(\boldsymbol{\theta})}}{\partial \boldsymbol{\theta}}. \tag{3.18}$$

The methods that aim to find the optimal policy in such a way are called policy gradient methods. However, this computation is generally intractable as we are trying to differentiate a function over a parameter when the function itself is conditioned on this parameter. The REINFORCE algorithm introduced by

Ronald J. Williams try to solve this problem based on mathematical simplifications. Namely, it is possible to show that under certain conditions equation (3.18) can be approximated as follows [113]

$$\Delta_{(\boldsymbol{\theta})} J_{(\boldsymbol{\theta})} \approx \sum_{t \geq 0} r_t^{(\tau)} \Delta_{(\boldsymbol{\theta})} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{x}_t) \tag{3.19}$$

This equation can be implemented in an iterative Monte Carlo approach which delivers to the REINFORCE algorithm described in the Algorithm 1 from [113], where v_t is the unbiased sample estimate of the $Q-$function $Q_{\pi_{\boldsymbol{\theta}}}(\mathbf{x}_t, \mathbf{a}_t)$, and α is the step size. With respect to the $Q-$function, we next define it when following the policy $\pi_{\boldsymbol{\theta}}$ for subsequent actions

$$Q_{\pi_{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{a}) = E[\sum_{t \geq 0} \gamma^t r_t^{(\tau)} | \mathbf{x}_0 = \mathbf{x}, \mathbf{a}_0 = \mathbf{a}]. \tag{3.20}$$

Algorithm 1 REINFORCE

Init $\boldsymbol{\theta}$ arbitrary

for each eposide $\{(\mathbf{x}_0, \mathbf{a}_0, r_1), \ldots, (\mathbf{x}_t, \mathbf{a}_t, r_{t+1}, , \ldots)\} \sim \pi_{\boldsymbol{\theta}}$ **do**
  for $t = 1$ to $T - 1$ **do**
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \Delta_{(\boldsymbol{\theta})} \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{a}_t) v_t$
  end for
end for

The REINFORCE method is in the class of so called *direct policy search algorithms*, as it does not make use of value functions [111]. However, this algorithm has high variance for a number of reasons. Namely, this is caused by the fact that in each experiment of Monte Carlo the rewards may vary a lot. Also, the attribution of the reward to each specific state-action in the trajectory is complicated, as REINFORCE averages out these rewards across the trajectory. Thus, the reward may be due to only a specific good state-action that can possibly hide a suboptimal behaviour across all other state-action pairs. In an attempt to solve these problems different versions of the REINFORCE algorithm have been introduced, including the change of the objective to cumulated future reward or discounted cumulated future reward,

and the inclusion of a baseline [113].

In the REINFORCE algorithm the policy function $\pi_{\boldsymbol{\theta}}$ is stochastic, however under certain conditions it is possible to derive a deterministic version of it. Namely, let's define with a $\pi_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{A}$ the deterministic policy that maps states to actions, if then the first derivative of this policy function ($\Delta_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{x})$) and the first derivative of its related Q function ($\Delta_a Q_\pi(\mathbf{x}, \mathbf{a})$) exist, it is possible to rewrite equation (3.18) as (Theorem 1 from *Silver et al.* [206])

$$\Delta_\theta J_{\boldsymbol{\theta}} = \mathrm{E}[\Delta_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{x}) \Delta_{\mathbf{a}} Q_\pi(\mathbf{x}, \mathbf{a})|_{\mathbf{a}=\pi_{\boldsymbol{\theta}}(\mathbf{x})}]. \tag{3.21}$$

More so, Theorem 2 from *Silver et al.* [206] shows that under certain conditions, when the variance of the stochastic policy gradient tends to zero, then the deterministic policy gradient is equivalent to the stochastic policy gradient under these limiting conditions. A key advantage over using the deterministic version over the stochastic version is computational, as the deterministic version does not require to integrate over all actions.

To get better estimates of the Q-function, one could replace the sample estimates with a neural network as well, yielding to what is known as the critic network $Q_{\boldsymbol{\theta}^Q}$. This, together with a policy network for $\pi_{\boldsymbol{\theta}^\pi}$ yields to an actor-critic framework. More so, to improve stability in the updates of the parameters, one could differentiate between these two networks and their target variants, the latter updating the parameters only intermittently and/or softly via moving averages. These *soft updates* deliver to the Deep Deterministic Policy Gradient algorithm (DDPG) [206]. DDPG is a model-free, off-policy, actor-critic algorithm based on the policy-gradient theorem which has been very successful in many applications. The algorithm updates the actor parameters via a sample approximation of $\Delta_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$, given by

$$\widehat{\Delta_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}} = \sum_i \Delta_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{x}_i) \Delta_{\mathbf{a}} Q_\pi(\mathbf{x}_i, \mathbf{a}_i)|_{\mathbf{a}_i=\pi_{\boldsymbol{\theta}}(\mathbf{x}_i)}. \tag{3.22}$$

3.3 Overview of Portfolio Optimisation

In this section we discuss the portfolio optimisation problem. We start by formulating it as a single period utility maximisation problem following *Scherer and Winston* [207]. We then discuss other common approaches which are not utility based. Finally, we consider recent Deep Learning approaches for portfolio optimisation. We conclude with a dynamic formulation of the problem that allows to use reinforcement learning techniques. The related literature has been surveyed in Section 2.3 to which we refer the reader. This section presents the theoretical background for Chapter 6.

3.3.1 Portfolio Optimisation and Utility Function

Portfolio optimisation is the process of finding the optimal allocation of wealth to different investments, given some objective and constraints. We are going to call $U(W_t)$ the utility function of a representative investor with wealth W_t at the end of period t, with $W_t = \boldsymbol{w}_t'(1+\boldsymbol{r}_t)$ where $\{\boldsymbol{r}_t\}_{t\geq 0}$ is assumed to be a vector of stationary stochastic processes representing asset returns, while $\boldsymbol{w}_t$ represents the vector of allocation weights over n assets chosen by the agent at the beginning of the period. The optimal single period asset allocation problem can be formulated as

$$\max_{\boldsymbol{w}_t\in\Omega} E[U(W_t)] = \max_{\boldsymbol{w}_t\in\Omega} \int\cdots\int U(\boldsymbol{w}_t'(1+\boldsymbol{r}_t))dF(\boldsymbol{r}_t) \tag{3.23}$$

where Ω is the set of permissible portfolios, while $F(\boldsymbol{r}_t)$ represents the joint cumulative distribution function of asset returns. The expected utility can be approximated via a Taylor Series around a root point, $\bar{W}$. The approximation converges to the true expected utility function for all levels of wealth when we are faced with an exponential or polynomial utility, while for power utility it converges only if the wealth is within a given range [208, 209, 210]. Hence, and assuming conditions for equality holds, we get

$$E[U(W_t)] = \sum_{k=0}^{\infty} \frac{U^{(k)}(\bar{W})}{k!} E[(W_t - \bar{W})^k] \tag{3.24}$$

where $U^{(k)}(\bar{W})$ represents the $k-th$ derivative of the utility function evaluated at the root point $\bar{W}$, and can be interpreted as the investor's preference (or aversion) towards the $k-th$ moment of the distribution. Notably, as observed by *Jondeau and Rockinger* [210] and shown by *Scott and Horvath* [211], under the assumptions of strict consistency for moment preferences, decreasing absolute risk aversion at all levels of wealth, and positive marginal utility, $U^{(k)}(\bar{W})$ is strictly negative for even k and strictly positive for odd k.

In case the root point $\bar{W}$ coincides with the expected wealth, we have the following fourth order Taylor approximation:

$$E[U(W_t)] \approx U(\mu_t^p) + \frac{1}{2}U^{(2)}(\mu_t^p)var_t^p + \frac{1}{6}U^{(3)}(\mu_t^p)sk_t^p + \frac{1}{24}U^{(4)}(\mu_t^p)\kappa_t^p. \quad (3.25)$$

where we define expected returns, variance, (un-standardised) skewness and kurtosis in the following ways: $\mu_t^p = E[\boldsymbol{w}_t'\boldsymbol{r}_t]$, $var_t^p = E[(\boldsymbol{w}_t'\boldsymbol{r}_t - \mu_t^p)^2] = E[(W_t - \bar{W})^2]$, $sk_t^p = E[(\boldsymbol{w}_t'\boldsymbol{r}_t - \mu_t^p)^3] = E[(W_t - \bar{W})^3]$, $\kappa_t^p = E[(\boldsymbol{w}_t'\boldsymbol{r}_t - \mu_t^p)^4] = E[(W_t - \bar{W})^4]$.

Alternatively, one can take the Taylor expansion of expected utility around $\bar{W} = 0$, yielding to the fourth order Maclaurin expansion:

$$\begin{aligned} E[U(W_t)] \approx & U(0) + U^{(1)}(0)M_1(W_t) + \frac{1}{2}U^{(2)}(0)M_2(W_t) + \\ & \frac{1}{6}U^{(3)}(0)M_3(W_t) + \frac{1}{24}U^{(4)}(0)M_4(W_t) \end{aligned}, \quad (3.26)$$

where $M_i(W_t)$ is the moment generating function. Conditions for sign identification of the relations between the different moments and the utility function can be found in *Scott and Horvath* [211].

Mean Variance Optimisation as a Special Case. Markowitz's theory suggests that two dimensions matter when picking up a portfolio that maximises the utility of an investor: expected return and variance of the portfolio. Thus, the problem of portfolio selection boils down to a trade-off between the mean return of a portfolio μ_t^p and its variance var_t^p. Therefore, yielding the following

optimisation problem:

$$\max_{\boldsymbol{w}_t \in \Omega} E[U(W_t)] = \mu_t^p - \frac{\lambda}{2} var_t^p = E[\boldsymbol{w}_t' \boldsymbol{r}_t] - \frac{\lambda}{2} E[(\boldsymbol{w}_t' \boldsymbol{r}_t - \mu_t^p)^2] \tag{3.27}$$

where λ is a positive parameter representing investor aversion towards risk, in this framework related to the second moment only of the return distribution.

It now emerges that equation (3.27) relates to equations (3.25) and (3.26) when we drop dependence of the utility function to the third and fourth moment of the distribution. This is legitimate in two circumstances: when the utility function is quadratic, and/or when returns follow a multivariate elliptical distribution. A critique raised against the quadratic utility assumption is that it provides unintuitive implications when returns rise above a critical value. In particular, investors maximising a quadratic utility function after a critical value prefer less return to more return [207].

On the statistical properties of asset returns, *Chamberlain* [212] showed that when these follow a multivariate elliptical distribution (e.g.: Normal, Student-t and Levy distributions), the mean-variance approximation is exact for all possible utility functions. Empirically, *Kroll et al.* [213] and *Jondeau and Rockinger* [210] found that the mean-variance provides a good approximation only under moderate non-normality.

Approaches to Higher Moments. When an investor is faced with two investment options exhibiting the same first and second moments, it comes naturally for the investor to consider higher order moments. A rational investor would certainly prefer higher skewness and lower kurtosis. Therefore, the mean-variance approximation is no longer appropriate when higher moments matter and cannot be spanned by lower ones [207]. This is indeed the approach taken, for example, by *Harvey et al.* [214], where skewness and co-skewness are accounted for when considering different portfolios. In *Fabozzi et al.* [215], portfolio selection in the presence of higher moments is discussed, and they show that a fourth order Taylor expansion of a logarithmic utility function delivers an objective that penalises variance and kurtosis, while rewarding

expected returns and skewness. Finally, *Jondeau and Rockinger* [210] analyse the case for the first four central moments of the expected utility function in general and of the CARA (Constant Absolute Risk Aversion) in particular; meanwhile, in *Lai et al.* [216], a polynomial goal programming framework is developed to solve the mean-variance-skewness-kurtosis problem.

Transaction Costs. Transaction costs can be modelled in different ways. Following *Sun et al.* [217] we can model linear transaction costs such that when taking allocation weights $\boldsymbol{w}_t$ from position $\boldsymbol{w}_{t-1}^+$ the cost incurred is

$$C_t = C(\boldsymbol{w}_t, \boldsymbol{w}_{t-1}^+) = \mathbf{c}'|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}^+| \tag{3.28}$$

where $\boldsymbol{w}_{t-1}^+$ is the pre-rebalance (end of period) asset position (i.e., $\boldsymbol{w}_{t-1}^+ = \frac{\boldsymbol{w}_{t-1} \circ (1+\boldsymbol{r}_t)}{\boldsymbol{w}_{t-1}'(1+\boldsymbol{r}_t)}$), $|\cdot|$ is the element-wise absolute value and $\mathbf{c}$ is the vector of transaction costs; thus, equation (3.28) induces a penalisation term on rebalancing.

Estimating Statistics. In order to maximise the approximated utility function in equations (3.25), (3.26) or (3.27), it is first necessary to compute an estimate for the statistics of interest, or more generally of the joint distribution function of asset returns $F(\boldsymbol{r}_t)$. Different approaches are available in the literature, starting from the simple use of historical sample moments, using moving average or exponential moving averages, or adopting time series models for the purpose. For example, autoregressive models can be estimated to provide next period estimates of the first and second moment of asset returns. The estimation of these statistics is tantamount to the performance, as errors propagate into the optimisation problem with the consequence of injecting noise in the asset allocation step. In Chapter 6 we provide estimates of the relevant statistics for the asset allocation strategies with two approaches, the first of which applies simple sample estimates using expanding windows. In the second approach we use the output of the Deep Dynamic Factor Model (D^2FM) introduced and elaborated in Chapter 4.

3.3.2 Other Approaches to Portfolio Optimisation

We now present other approaches to the problem of optimal asset allocations that we will use and empirically evaluate in Chapter 6. All of them require an estimate for the relevant statistics characterising some aspects of the distribution of next period asset returns.

Minimum Variance. Minimum variance is similar in spirit to the mean variance optimisation framework, but assumes that investors care only about risk. Hence, the objective function is:

$$\min_{\boldsymbol{w}_t \in \Omega} E[(\boldsymbol{w}_t' \boldsymbol{r}_t - \mu_t^p)^2]. \tag{3.29}$$

A comparison between mean variance and minimum variance signals that a minimum variance investor has a lower expected return profile. Nevertheless, empirically high volatility and high beta stocks have performed worst than low volatility and low beta stocks in the U.S. market [218]. Additionally, one can think about the minimum variance as a special case of the mean-variance where prices's stochastic processes are martingales, hence making expected returns zero.

Risk Parity. The risk parity portfolio consists in allocating wealth to the assets so that they contribute equally to the variance of the portfolio; while mean-variance generally leads to concentrated positions in a few assets, the risk parity approach favours diversification [219]. The objective function in this setting is

$$\min_{\boldsymbol{w}_t \in \Omega} \sum_{i=1}^{n} [w_{t,i} - \frac{(\sum_{i=1}^{n} \sigma_i(\boldsymbol{w}_t))^2}{(\Sigma_t \boldsymbol{w}_t)_i N}]^2, \tag{3.30}$$

where Σ_t is the variance covariance matrix of asset returns $\boldsymbol{r}_t$, while $(\Sigma_t \boldsymbol{w}_t)_i$ is the $i-th$ element of the vector of the product of this matrix with the asset weights. Finally, $\sigma_i(\boldsymbol{w}_t) = \frac{w_{t,i}(\Sigma_t \boldsymbol{w}_t)_i}{\sqrt{\boldsymbol{w}_t' \Sigma_t \boldsymbol{w}_t}}$.

Hierarchical Risk Parity. Hierarchical risk parity, as introduced by *De Prado* [41, 220], deals with three major concerns of quadratic optimisers in general and the Markowitz framework in particular: instability, concentration

and under-performance. *De Prado* argues that in any optimisation framework that involves the inversion of the variance-covariance matrix, the higher is the correlation among asset returns, the more unstable the solution, notwithstanding the higher the need for diversification in such a situation. This is what *De Prado* [41] calls the Markowitz' curse, and to deal with the problem he introduces an algorithm with three steps: Hierarchical Tree Clustering, Quasi-Diagonalisation and Recursive Bisection. The Hierarchical Tree Clustering step at first converts a correlation matrix ρ_t into a correlation distance matrix D_t, such that element $D_{t,i,j} = \sqrt{0.5(1-\rho_{t,i,j})}$. Then another distance matrix is constructed to measure the closeness in similarity of two assets with respect to all other assets in Euclidean metric. This measure takes the form $\bar{D}_{t,i,j} = \sqrt{\sum_{k=1}^{d}(D_{t,k,i} - D_{t,k,j})^2}$. Finally, clusters are defined by taking the two elements at the minimum of this distance matrix. The procedure is repeated by recursively dropping the clustered elements so as to construct a dendrogram. The Quasi-Diagonalisation step consists in a rearrangement of the covariance matrix such that similar assets are placed together, and dissimilar ones are placed far apart. Finally, allocations are provided by the Recursive Bisection step. Namely, all weights are initialised to 1. Then, starting from the topmost cluster, we compute the variance of the sub-clusters as $\tilde{V}_{t,i} = \boldsymbol{w}_t' V_{t,i} \boldsymbol{w}_t$, where $V_{t,i}$ with $i = 1,2$ is the variance of child cluster i, while $\boldsymbol{w}_t = \frac{diag(V_t)^{-1}}{trace(diag(V_t)^{-1})}$. Successively, a split factor is computed as $\alpha_t = 1 - \frac{\tilde{V}_{t,1}}{\tilde{V}_{t,1}+\tilde{V}_{t,2}}$. Finally, weights in child cluster 1 are scaled by α_t, while those in child cluster 2 by $1-\alpha_t$. The process is repeated until leaf nodes. A detailed implementation of the algorithm can be found in the original source [41, 220].

3.3.3 Deep Portfolio Optimisation

Recently, a number of papers have proposed the use of deep learning techniques to solve the asset allocation problem [see 23, 24, 25, 26, 27, for example]. In such a framework, asset allocation weights are given by a parametrised deep

learning model of the following form:

$$\boldsymbol{w}_t = DNN(\mathbf{x}_t; \boldsymbol{\theta}) \tag{3.31}$$

where $DNN(\mathbf{x}_t; \boldsymbol{\theta})$ stands for a generic deep neural network with parameters $\boldsymbol{\theta}$ and inputs $\mathbf{x}_t$. One difference in this approach compared to the aforementioned is the combination of the prediction and optimisation step in a unique one. Indeed, the method does not require the user to plug in any estimate of key statistics for future returns distribution, but rather only the history of returns together with any other feature that the user deems relevant, and which we collect in $\mathbf{x}_t$ (on top of the objective function to optimise). Another important difference with respect to the static myopic approaches discussed thus far is the dynamic aspect of this methodology, and its relation to Dynamic Programming and Reinforcement Learning. The DNN in equation (3.31) can be seen as a parametrised policy function solving the dynamic asset allocation problem that we will discuss in next section. Nevertheless, the link is clear only when cumulative rewards in terms of profits are specified as objective function (or the average returns as in *Noguer and Srivastava* [24]), but when other types of loss are designed, like the Sharpe ratio, such connection is no longer sharp [23].

In order to optimise the parameters $\boldsymbol{\theta}$, we need to define an objective function (or loss function). In *Zhang et al.* [23] the Sharpe ratio is used, i.e., $-L(\boldsymbol{\theta}) = \frac{\hat{\mu}_t^p(\theta)}{\hat{\sigma}_t^p(\boldsymbol{\theta})}$, while *Noguer and Srivastava* [24] use $-L(\boldsymbol{\theta}) = \hat{\mu}_t^p(\boldsymbol{\theta})$. Here, we set the minus in front of the loss function $L(\boldsymbol{\theta})$, as this one is generally minimised. Estimates of the statistics are computed over a given trading period; thus, we have $\hat{\mu}_t^p(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t)'\boldsymbol{r}_t)$ and $\hat{\sigma}_t^p(\boldsymbol{\theta}) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t)'\boldsymbol{r}_t - \hat{\mu}_t^p(\boldsymbol{\theta}))^2}$, where $\pi_{\boldsymbol{\theta}}(\mathbf{x}_t)$ are the asset allocation decisions taken for period t, which are function of the neural network input features summarised in the vector of relevant states $\mathbf{x}_t$, and its parameters $\boldsymbol{\theta}$ (i.e., $\pi_{\boldsymbol{\theta}}(\mathbf{x}_t) := DNN(\mathbf{x}_t; \boldsymbol{\theta})$). In both methods, the output layer is a softmax function so that weights are larger than zero and sum up to 1, thus, imposing a long-only fully invested constraint. Generalised gradient methods on the loss function $L(\boldsymbol{\theta})$ allow us to ameliorate

the parameters θ and the conditional allocation weights $\boldsymbol{w}_t = \pi_{\boldsymbol{\theta}}(\mathbf{x}_t)$ with respect to the specified loss function.

3.3.4 Dynamic Framework

Up to now, and with the exception of the previous section, we have discussed the static version of the portfolio optimisation problem. Namely, we have assumed a single period framework. We now proceed by formulating the dynamic version of the problem within a self-financing framework, thus without accounting for consumption and income effects. Before doing this, let's define the MDP faced by the investor. We assume at any given point in time t the investor observes a vector of states $\mathbf{x}_t$, conditioned on which he takes action $\boldsymbol{w}_t = \pi(\mathbf{x}_t)$. Upon taking this action, the investor incurs into an instantaneous cost C_t defined in equation (3.28) which cumulates with the expected reward. The latter is a function of the allocation strategy ($\boldsymbol{w}_t$) and the next period returns' distribution $\boldsymbol{r}_{t+1}$ which we assume to be both fully characterised by the current states $\mathbf{x}_t$. The system then evolves according to a given transition probability kernel $\mathcal{P}$ from state $\mathbf{x}_t$ to the new state $\mathbf{x}_{t+1}$, and so on. Thus, similarly to *Sun et al.* [217] we can write the following Bellman equation[8]

$$J_t(\mathbf{x}, \pi) = \max_{\pi} \{E[G(\mathbf{x}_t, \pi) + J_{t+1}(\mathbf{x}_{t+1}, \pi) | \mathbf{x}_t = \mathbf{x}]\}, \tag{3.32}$$

with the instantaneous reward defined as $E[G(\mathbf{x}_t, \pi)|\mathbf{x}_t = \mathbf{x}] = E[C(\pi(\mathbf{x}_t), \boldsymbol{w}^+_{t-1}) + U(\pi(\mathbf{x}_t)'(1+\boldsymbol{r}_{t+1}))|\mathbf{x}_t = \mathbf{x}]$, where we have collected the previous period weights in the state vector $\mathbf{x}_t$ together with all other relevant information to ensure Markovianity. This formulation of the problem allows us to apply the techniques discussed in Section 3.2.5.[9] Indeed, in the dedicated section of the literature review chapter (see 2.3), we introduced the formulation of the dynamic portfolio

[8] *Sun et al.* [217] use this formulation in the context of optimal portfolio rebalancing, we borrow their formulation and adapt it to our context.

[9] A note of caution, however, might be made in case one would like to apply deterministic policy gradient methods (see 3.2.5) and at the same time (s)he wishes to specify transaction costs as in equation (3.28). Indeed, in this case the first derivative of the Q function ($\Delta_a Q_\pi(\mathbf{x}, \mathbf{a})$) does not exist when the policy suggests not to rebalance (i.e., $\pi(\mathbf{x}_t) = \boldsymbol{w}^+_{t-1}$).

optimisation problem as an MDP and we surveyed the related literature.

Finally, as discussed in *Back* [110], when the return vector $\boldsymbol{r}_t$ is indipendent and identically distributed (*i.i.d.*) and we are in front of a linear risk tolerance utility function, then there is not much new in the dynamic formulation of the problem as opposed to the single period formulation discussed previously.

3.4 Explainable Deep Learning

Deep learning models can be highly nonlinear with many interaction entities; thus, behaving akin black boxes to humans. Where a black box is a type of model whose workings are either proprietary or known but uninterpretable to humans [221, 28]. Thus, assuming the human user has access to a Deep Learning model, (s)he could still be unable to interpret it. In order to solve this problem many scientists have focused on the design of tools to explain their models or on building models that are inherently interpretable. This has given rise to a new branch of AI called *eXplainable AI (XAI)* [29]. In the literature different surveys of XAI with related taxonomies are available [221, 222, 223, 30], and we intend to conform to them in the following.

3.4.1 Interpretability and Explainability

While these two concepts are sometime used in an interchangeable manner [224], *Rudin* [28] makes a clear distinction among the two and in this work we intend to conform to this. Namely, a model is interpretable when its working mechanisms are intrinsically comprehensible to humans, as it is the case for linear regressions and decision trees for example [221];[10] on the other side explainability refers to the possibility of building a post hoc trustworthy interface between the human and the model that makes some or all aspects of the model comprehensible to humans [28, 30].

To formalise the model explanation problem we make use of the following definition from *Guidotti et al.* [221].

[10]Sometime this feature is also expressed as transparency [30].

Definition 1 (Model Explanation Problem). *Given a black box predictor $G_{\boldsymbol{\theta}}(\cdot)$ and a set of instances $\mathcal{X}$, the model explanation problem consists in finding an explanation $\varepsilon x \in \mathcal{E}$, belonging to a human interpretable domain $\mathcal{E}$, through an interpretable global predictor $c_g = f(G_\theta(\cdot), \mathcal{X})$ derived from the black box $G_{\boldsymbol{\theta}}(\cdot)$ and the set of instances $\mathcal{X}$ using some process $f(\cdot,\cdot)$. An explanation $\varepsilon x \in \mathcal{E}$ is obtained through c_g, if $\varepsilon x = \varepsilon x_g(c_g, \mathcal{X})$ for some explanation logic $\varepsilon x_g(\cdot,\cdot)$, which reasons over c_g and $\mathcal{X}$.*

This definition is very general and it includes the case where an explanation logic is needed to process the interpretable predictor c_g. As it emerges from this definition, interpretability is a characteristic of the model, while explainability refers to the sequence of actions that goes from the selection of an interpretable model to the application of the explanation logic to map the output of the interpretable model to the human interpretable domain $\mathcal{E}$.

The issue of directly building interpretable models instead of looking for a post-hoc intepretable representation of the original model is formalised in the definition of the *Transparent Box Design Problem* from *Guidotti et al.* [221].

Definition 2 (Transparent Box Design Problem). *Given a training dataset $D = \mathcal{X}, \mathcal{Y}$, the transparent box design problem consists in learning a locally or globally interpretable predictor c from D. For a locally interpretable predictor c, there exists a local explanator logic εx_l to derive an explanation $\varepsilon x_l(c, \mathbf{x})$ of the decision $c(\mathbf{x})$ for an instance $\mathbf{x}$. For a globally interpretable predictor c, there exists a global explanator $\varepsilon x_g(c, \mathbf{x})$ to derive an explanation $\varepsilon x_g(c, \mathbf{x})(c, \mathcal{X})$.*

In general one is rarely interested in all possible $\varepsilon x \in \mathcal{E}$, and most often researchers are concerned only about a subset of $\mathcal{E}$. The definition of this subset is problem specific, as well as the design of explanability tools and that of interpretable models. In the context of DFMs discussed in 3.1.2, *Banbura and Modugno* [39] introduce the concept of *news* as an explanation of the change in the conditional prediction of a target variable given the new information set. In this explanation context and within their DFM framework, those quantities

are directly computable and thus their model can be considered interpretable in this sense. In *Rudin* [28], the author explicitly pushes scientists to dedicate their efforts to the development of *inherently interpretable* models, which we link to the *Transparent Box Design Problem* of *Guidotti et al.* [221], and which we identify in the DFM of *Banbura and Modugno* [39]. After discussing the many technical obstacles in building inherently interpretable models, *Rudin* [28] recognise the difficulty in collecting literature on interpretable machine learning models. Indeed, the authors comment that often the keyword *interpretable* is neither mentioned in the title nor in the body of the text. In particular, works on interpretable machine learning mentioned in the paper build interpretable models by including constraint in the optimisation and/or using dimensionality reduction to interpretable dimensions [225, 226, 227, 228, 229]. In particular, in *Gallagher et al.* [227], the authors analyse brain-wide electrical spatiotemporal dynamics in a compressed space in order to understand depression vulnerability. Also in this sense a DFM can be considered interpretable, as it maps a large dataset to a lower dimensional space of common components that aim to represent features of the Busyness Cycle. For these reasons, in Chapter 4 we introduce a new modelling framework that extends the DFM from *Banbura and Modugno* [39] with some of the deep learning techniques introduced in 3.2. Notably, the new framework remains interpretable with respect to the different aspects discussed.

3.4.2 Other Taxonomies

Explainability methods are *model specific* when they leverage the structure of the model being explained to make its output understandable to humans. For DL models, the Integrated Gradient [230] and the Deep SHAP [231] represent model specific examples. As such, those approaches have the drawback of being applicable only to the type of models for which they were designed. On the contrary, *model agnostic* approaches are more flexible and general, as they are designed to explain general black-box types of predictors. Examples in this category are LIME [232], and Sampled and Kernel SHAP [233, 234, 140, 231].

When measuring the importance of a feature we can use methods that focus on a single instance or on the entire dataset. The former are referred as *local methods*, while the latter are called *global methods*. However, if we run a local method on an entire dataset, we could of course compute some metrics from it to derive global measures.

Surrogate methods for explainability consist of implementing one or more interpretable models to explain the original black box predictions. Arguably, this approach generates wrong explanations or it is not completely faithful to the original ones. Indeed, if this were not the case, one could have in principle used the interpretable model [28]. Conversely, *visualisation methods* do not use a different model, but rather an algorithm to explain the black box model through visualisation maps. Feature attribution methods are in this category, and we will next discuss one of them.

3.4.3 Feature Attribution Methods: Shapley Values and the Choice of a Baseline

Feature attribution methods are used to indicate how much each feature contributes to the prediction for a given example. As such they can be classified as a local explanation method, although one could compute their global version by taking some statistic over the local explanations (e.g., average or median). In an attempt to unify notation in [231, 235] with the one of this thesis, we define them as follows

Definition 3. *[Additive feature attribution methods] They have an explanation model that is a linear function of binary variables:*

$$\varepsilon x_l(\mathfrak{m}, \mathbf{x}) = \phi_0(\mathfrak{m}, \mathbf{x}) + \sum_{i=1}^{K} \phi_i(\mathfrak{m}, \mathbf{x}) z_i, \tag{3.33}$$

where $\boldsymbol{z} \in \{0,1\}^K$*, K is the number of input features, and* $\phi_i(\mathfrak{m}, \mathbf{x}) \in \mathbb{R}$*.*

The variable z_i represents a feature being observed when it takes the value of 1 or unknown when it takes the value of 0, while $\phi_i(\mathfrak{m}, \mathbf{x})$ represents the

$i-$th feature attribution value based on model $\mathfrak{m}$ and input $\mathbf{x}$. This definition implies the need of the construction of a mapping $h_{\mathbf{x}} : \{0,1\}^K \to \mathcal{X}$, such that $h_{\mathbf{x}}(\mathbf{z}) = \mathbf{x}$. Moreover, the model $\mathfrak{m}$ could be the original model itself ($\mathfrak{m} = G_\theta$) or an interpretable version of it ($\mathfrak{m} = c_g$ from Definition 1).

A theoretically grounded feature attribution method that uses directly the original model (i.e., $\mathfrak{m} = G_{\boldsymbol{\theta}}$) is provided by the Shapley values and their approximations [231, 236]. In particular, Theorem 1 from *Lundberg and Lee* [231] shows that Shapley values are the unique possible explanation model that follows Definition 3 while satisfying three desirable properties: *local accuracy, missingness and consistency.* Local accuracy establishes that the sum of the feature attributions equates the output of the model we wish to explain. Missingness states that missing features (i.e., $z_i = 0$) have no importance. Consistency ensures that when changing the model to a model where a given feature has more importance, this will not decrease the attribution calculated for the feature.

For a neural network, $G_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$, and K input features, the contribution of feature j calculated according to the Shapley value for input $\mathbf{x} = [x_1, x_2, \ldots, x_K]$ is given by

$$\phi_j(G_{\boldsymbol{\theta}}, \mathbf{x}) = \sum_{S \subseteq P \setminus \{j\}} \frac{|S|!(|P| - |S| - 1)!}{|P|!} \left(G_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{S \cup \{j\}}) - G_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_S) \right), \tag{3.34}$$

where P is the collection of all feature indices, element i of vector $\tilde{\mathbf{x}}_S$ is given by $\tilde{x}_{S,i} = x_i \mathbb{1}_{\{i \in S\}} + b_i \mathbb{1}_{\{i \notin S\}}$ (similarly for $\tilde{\mathbf{x}}_{S \cup \{j\}}$), and b_i is the baseline value for feature i. The *baseline* models the missingness of a feature, i.e., it replaces that feature when it is absent (i.e., $z_i = 0$). As argued by *Sturmfels et al.* [237], the concept of missingness is not well defined or explored in ML models. Alternatively, feature(s) can be removed from the model via marginalisation, thus assuming a distribution for b_i. Nevertheless, the standard practice in setting up a baseline is to assign a vector of zeros for all features. However, this choice might provide wrong interpretations and even give zero importance

to significant features.

Whereas as the number of features increases, calculating Shapley values becomes expensive because the computational cost is exponential with respect to the number of features. Hence, approximation methods have been developed, of which the most commonly used ones in Deep Learning are Deep Taylor Decomposition [238], Integrated Gradient [230], Deep SHAP [231], and Deep LIFT [239]. Other model agnostic methods are Sampled and Kernel SHAP [233, 234, 140, 231]. All of these methods require the choice of a baseline.

The simplest choice of a baseline is the *zero* vector baseline [240, 230, 239, 236], which coincides with the *average* vector baseline when features are standardised. However, this choice could be misleading. For instance, consider a feature in a model that is most significant when its value is zero. Now, if we compute the Shapley values on this model with the zero baseline, the importance of that will be zero. One way of addressing the zero-baseline insensitivity problem is to use the maximum distance baseline (mdb) method [237]. This baseline consists of taking the furthest observation from the current one in an L^1 norm. However, this approach unequivocally creates incoherent justifications for the interpretations provided by the model due to the dependence of the baseline to each of the instances being explained.

Alternatively, one can sample a baseline from a multivariate distribution such as Uniform [237] or Gaussian [241]. This approach can be improved by considering a sample of baselines and to average the attributions computed over each baseline [241, 231, 242, 243, 244]. Another form of sampling is the one performed using the input data. Hence, one can use the underlying inputs' empirical distribution [231, 244, 237]. We denote this as the $p_{\mathcal{X}}$ baseline method. However, the $p_{\mathcal{X}}$ baseline increases the computational cost of estimating feature attributions linearly with respect to the number of draws. Moreover, this choice of baseline does not allow the setting of a reference value on the model output when computing the Shapley values. This is important when decisions are taken with respect to a specific value of the model. This generalises also to the

other baselines described before.

For the Deep Taylor Decomposition approach, the baselines are chosen using Layer-wise Relevance Propagation [137] or Pattern Attribution [245]. The drawback of these baseline search methods is the non applicability to other attribution methods, such as exact Shapley values or Shapley sampling.

Since the definition of baselines is fundamental to the correct interpretation of Shapley values [246], in chapter 5 we analyse this problem in detail and propose a novel approach to search for them in MLPs.

3.4.4 Measure of Explainability Power

The evaluation of explainability methods from a quantitative perspective is difficult due to the lack of a clear definition of what is a correct explanation [247, 230, 236]. Many extrinsic evaluation of explainability power have been developed and are based on measuring the effect on a model when removing the most important feature, as was identified by a feature attribution method. Some authors quantify this effect by measuring the difference in performance, and others by measuring the difference in prediction value. The intuition behind these methods is that if the feature attribution method correctly identifies the most important feature, then when this feature is removed, the model performance should decrease more (or the prediction value should deviate more) than when a less important feature is removed. In this thesis, we will use these evaluation methods to compare the choice of baselines.

Among these methods, two emerge in particular. The so-called RemOve And Retrain (ROAR) [248], that consists in, given a model, first identifying the most important feature on a per example basis and then retraining the model with the values of the features substituted with their respective averages, and measuring the difference in performance between the two models. *Ancona et al.* [249] propose an alternative measure of local evaluation. Given a model, one must first identify the most important feature for each example, then remove this feature by substituting it with its baseline value and measuring the deviation in prediction. These deviations are then averaged to obtain a

single value. A limitation of these evaluation methods is that, since a feature is removed one at the time, these measures may be misled by potential feature correlations.

Chapter 4

Deep Dynamic Factor Models (D^2FMs) as Interpretable Models

In this chapter we introduce and evaluate a new Deep Learning (DL) framework which is constructed to embrace the suggestion of *Rudin* [28] to build *inherently interpretable* models, and the *Transparent Box Design Problem*[1] of *Guidotti et al.* [221]. In fact, we merge the DL literature on autoencoders with the econometric one on Dynamic Factor Models (DFMs) and *Nowcasting*.[2] The combination will comprise a family of models – which we label Deep Dynamic Factor Models (D^2FMs or DDFMs) – and is general enough to account for many functional forms and datasets, while mantaining interpretablility aspects of DFMs.

In particular, by embedding autoencoders in a dynamic nonlinear factor model structure to tackle financial and macroeconomic problems, we provide generalisation of linear factor models. Importantly, the equivalence between maximum likelihood estimation and minimisation of mean squared error in the presence of conditional Gaussianity [159], together with the Universal Approximation Theorem [188, 189, 190], allow us to conceptualise D^2FMs and the procedure adopted in estimating them as an efficient computational method to approximate the maximum likelihood estimates of nonlinear factor

[1] See Definition 2.

[2] Nowcasting is a contraction of the term *now* and *forecasting* and it refers to the prediction of the present, the very near future, and the very recent past state of an economic indicator.

models. Another contribution of this work is to show how to incorporate general patterns of missing data, jagged edges and mixed frequencies in this framework by extending gradient-based backpropagation methods for DL.

Our methodology scales better compared to (approximate) maximum likelihood methods for DFMs when the dataset contains many features and/or observations. To the best of our knowledge, this work is the first to adapt an autoencoder structure into a dynamic model with both factor dynamics and dynamic idiosyncratic components, in a state-space framework for real-time high dimensional mixed frequencies time-series data with arbitrary patterns of missing observations. The proposed D^{2}FM framework is very general and can in principle be applied to many different problems both in forecasting and in structural analysis, as it is done with DFMs. The model is designed to be in spirit as close as possible to DFMs.

Through a Monte Carlo simulation exercise and on a macroeconomic dataset we empirically demonstrate the ability of the model to outperform the standard DFM. For the Macroeconomic dataset we encode, using our model and the DFM, the full *McCracken and Ng* [250] FRED-MD dataset, a large macroeconomic database for the U.S. economy, which has been specifically designed for the empirical analysis of *big data*.

On the methodological aspect, this chapter connects with a number of works from the DL literature [200, 205, 202, 203, 201, 204, 251, among others] which has been covered in Section 3.2.4 of Chapter 3. We also connect with the literature that has been using DL techniques to estimate state space models. In particular, *Rangapuram et al.* [204] propose a recurrent neural network (RNN) structure that, given the input features, provides the latent states together with all the (time-varying) parameters of the state space model. Optimisation is carried out on the log-likelihood efficiently computed via the Kalman filter. They argue that, different from Deep AR [251] models, in their approach the target values are not used as inputs, thus making their model more robust to noise and able to handle missing data. We tackle those two issues differently.

For the former we apply a denoising approach, for the latter we use selection matrices to mask missing data on the output side, while the generative spirit of our model allows us to do conditional sampling in order to fill missing inputs during the training. Additionally, in all of the surveyed works mixed frequencies and variable specific (idiosyncratic) components are not taken into consideration. In our approach we propose a way to deal with both of these issues. The first we resolve by including restriction matrices in the emission equation à la *Mariano and Murasawa* [252]. The second is handled by designing an alternated optimisation scheme between common factors and idiosyncratic components by means of a Markov Chain Monte Carlo algorithm.

By observing that factor models can be thought of as a special case in the class of the dynamic autoencoder models, this work connects also with the large and influential literature on factor models in Economics. Since its onset, a key problem in the factor model literature has been that, due to the latency of the factors, maximum likelihood estimators cannot be derived explicitly. *Geweke* [83] and *Sargent and Sims* [84] for the frequency domain, *Engle and Watson* [253] and *Stock and Watson* [38] for the time domain have proposed algorithms for small dynamic latent models. The common drawback of all these proposed methods is that, in general, the maximum likelihood approach is impractical for datasets where the cross-section size is large. To solve this problem, *Forni and Reichlin* [254], *Stock and Watson* [85] and *Giannone et al.* [255] have proposed non-parametric methods based on principal component analysis to estimate the latent components with large cross-section data.[3] *Banbura and Modugno* [39] have provided a methodology to deal with any pattern of missing data, while allowing for mixed frequencies series and jagged edge issues in large datasets.[4] Their methodological framework, which has been introduced in Section 3.1, will be used as a building block and benchmark for the methodology introduced in

[3]Recently, *Doz et al.* [154] and *Barigozzi and Luciani* [37] showed that when the size of the cross-section tends to infinite the estimates obtained by a quasi-maximum likelihood approach are consistent, also when there is a weak cross-sectional correlation in the idiosyncratic components.

[4]*Jungbacker et al.* [256] provide efficient and fast treatment of missing data in the dynamic factor model estimation.

this chapter.

Finally, we connect to the econometric literature that has explored the extend of nonlinearities in macroeconomic data and proposed univariate and multivariate nonlinear time-series models [see 257, for a comprehensive literature review]. An important part of this literature has estimated dynamic latent models with nonlinearities, which are explicitly modelled through structural breaks, Markov switching regression or threshold regression [258, 259, 260, 261, 262].[5] The approach of *Bai and Ng* [264] is the closest in spirit to ours and an important early effort at including nonlinearities in factor models. In that paper, either principal components of nonlinear transformation of the data are estimated or nonlinear transformation of the factors are added to a linear factor model. Our methodology is more general. In fact, different from their procedure, our D^{2}FMs need not to assume a specific form of nonlinearity, either in the encoding or in the decoding map.[6]

4.1 Encoding in Economics

An overreaching idea in Macroeconomics, already present in the work of *Burns and Mitchell* [144], is that a few common forces can explain the joint dynamics of many macroeconomic variables. This stylised view of the economic data generating process informs the effort of economic modelling in – for example, the Real Business Cycle (RBC) and Dynamic Stochastic General Equilibrium (DSGE) literature – and is one of the very few robust facts in empirical macroeconomics, motivating the use of factor models [see 266, for example].

In Macroeconometrics, factor models were introduced by *Geweke* [83] and *Sargent and Sims* [84] and are the very first instance of *big data* in the field. Dynamic Factor Models (DFMs) deal with a large cross-section problem by applying a linear dynamic latent state framework to the analysis of economic

[5] Nonlinearities have been also modelled in structural factor models using DSGE models (Dynamic Stochastic General Equilibrium models) as in *Amisano and Tristani* [263] to detect regime switching in volatility.

[6] The deep learning literature refers to this as a shift from *feature engineering* to *architecture engineering* [see 265, for example].

time-series. The underlying assumption of these models is that there is a small number of pervasive, unobserved, common factors that stir the economy and inform the comovements of hundreds of economic variables. Economic times series are also affected by variable-specific (idiosyncratic) disturbances. These idiosyncratic disturbances can be due either to measurement error or to factors that are specific to some variables. A large body of empirical evidence produced using dynamic factor models, shows how a small number of factors – as many as two – can capture a dominant share of the variance of all the key macroeconomic and financial variables. This family of models has been applied intensively in Econometrics to different problems such as forecasting, structural analysis and the construction of economic activity indicators [see 85, 86, 87, 88, 89, 90, 91, 92, among others]. Factor models are robust and flexible models, also able to accommodate for missing observations, jagged patterns of data and mixed frequencies.[7] However, arguably their most important limitation is that they almost always assumed linear structure.

4.2 Autoencoders and Factor Models

Autoencoders are neural networks trained to map a set of variables into themselves, by first encoding the input into a lower dimensional representation and then decoding it back into itself.[8] The lower dimensional representation forces the autoencoder to capture the most salient features of the data. In constructing a nonlinear reflexive map that links the inputs back to itself via a lower dimensional space, autoencoders can be thought of as a nonlinear generalisation of PCA. Recently Deep AutoRegressive Networks [200] and Temporal Difference Variational Autoencoders [201] have been introduced to extend the static autoencoder framework to a dynamic environment.

On the other side, dynamic factor models for econometric time series are multivariate probabilistic models in which a vector of stochastic disturbances

[7] Jagged edges arise when there is a varying number of missing observations at the end of multiple time-series due to non-synchronous release dates.

[8] See Section 3.2.3 for additional details on autoencoders.

are transmitted via linear dynamic equations to the observed variables. They assume that a small number of stochastic unobserved common factors informs the comovements of hundreds of economic variables. In doing this, they combine two core ideas of macroeconomics: the Frisch-Slutsky paradigm, which assumes the economic variables to be generated by the stochastic components (the economic shocks) via usually linear dynamic difference equations [see 267, for example]; and the *Burns and Mitchell* [144] idea: that a few common disturbances explain most of the dynamics of all the macroeconomics variables, with a residual share due to idiosyncratic components. DFMs are similar in intuition to principal component analysis (PCA), but assume stochastic and dynamic structure that allows for their application to econometric times series.

In this section, we explore the deep connection between factor models and autoencoders to show that the dynamic formulation of autoencoders can be regarded as a nonlinear generalisation of dynamic factor models in the same way that standard autoencoders can be seen as generalisations of principal component analysis.

4.2.1 Latent Factor Models

Let us first introduce a general formulation of latent factor models with idiosyncratic components. We define $\mathbf{y}_t = (y_{t,1}, ..., y_{t,n})$ as the vector collecting the n variables of interest at time t, usually assumed to be the realisation of a vector stochastic process. A general latent factor model can be written as

$$\mathbf{y}_t = F_{\boldsymbol{\theta}_F}(\boldsymbol{f}_t) + \boldsymbol{\varepsilon}_t = \widetilde{\mathbf{y}}_t + \boldsymbol{\varepsilon}_t \ , \tag{4.1}$$

where $\boldsymbol{f}_t$ is an $r \times 1$ (for $r = dim(\boldsymbol{f}) << n = dim(\mathbf{y})$) vector of latent common stochastic components – i.e. the factors –, $\boldsymbol{\varepsilon}_t$ are idiosyncratic and unobserved stochastic error terms, and $F_{\theta_F}(\cdot)$ is a generic function mapping the unobserved factors into the observed variables. Usual assumptions are that $\boldsymbol{f}_t$ and $\boldsymbol{\varepsilon}_t$ are independent, with zero mean and finite variance (the variance of $\boldsymbol{f}_t$ is often assumed to be a diagonal matrix). For later reference, we indicate as $\widetilde{\mathbf{y}}_t$ the

projection of $\mathbf{y}_t$ into the factors $\boldsymbol{f}_t$. By also assuming a linear function $F_{\boldsymbol{\theta}_F}(\cdot)$, the model reduces to the standard linear factor model

$$\mathbf{y}_t = \boldsymbol{\theta}_F \boldsymbol{f}_t + \boldsymbol{\varepsilon}_t. \tag{4.2}$$

However, in general, $F_{\boldsymbol{\theta}_F}(\cdot)$ need not be linear and we can express the factor component of equation (4.1) as

$$\widetilde{\mathbf{y}}_t = F_{\boldsymbol{\theta}_F}(G_{\boldsymbol{\theta}_G}(\mathbf{y}_t)) = (F_{\boldsymbol{\theta}_F} \circ G_{\boldsymbol{\theta}_G})(\mathbf{y}_t) = (F_{\boldsymbol{\theta}_F} \circ G_{\boldsymbol{\theta}_G})(\widetilde{\mathbf{y}}_t + \boldsymbol{\varepsilon}_t), \tag{4.3}$$

where $G_{\boldsymbol{\theta}_G}(\cdot)$ is the function mapping the observables into the *code* $\boldsymbol{f}_t$ (encoding function), and $F_{\boldsymbol{\theta}_F}(\cdot)$ is the function mapping the factors back into $\mathbf{y}_t$ (decoding function), that is $\widetilde{\mathbf{y}}_t$. The connection between factor models and autoencoders is more evident in this form. In fact, the map in equation (4.3) can be regarded as a very general autoencoder. Linear factor models can be seen as a special case of factor models assuming both a linear encoding and a linear decoding function. It is worth noting that the model in equations (4.1) and (4.3), together with a specification of the dynamic transition equations, correspond to the static representation of a Dynamic Factor Model (see Section 3.1).

4.2.2 Dynamics in Factor Models

Thus far we have discussed the general structure of factor models by abstracting from the dynamics and focusing on the static map into lower dimensional factors. Dynamics is usually introduced in DFMs by assuming that both $\boldsymbol{f}_t$ and $\boldsymbol{\varepsilon}_t$ are generated by linear stochastic vector difference equations. For example, *Banbura et al.* [93] and *Banbura and Modugno* [39] consider a system specified as

$$\mathbf{y}_t = \boldsymbol{\theta}_F \boldsymbol{f}_t + \boldsymbol{\varepsilon}_t \ , \tag{4.4}$$

$$\boldsymbol{f}_t = \boldsymbol{B}_1 \boldsymbol{f}_{t-1} + \cdots + \boldsymbol{B}_p \boldsymbol{f}_{t-p} + \boldsymbol{u}_t, \qquad \boldsymbol{u}_t \overset{iid}{\sim} \mathcal{N}(0, \boldsymbol{U}), \tag{4.5}$$

$$\boldsymbol{\varepsilon}_t = \boldsymbol{\Phi}_1 \boldsymbol{\varepsilon}_{t-1} + \cdots + \boldsymbol{\Phi}_d \boldsymbol{\varepsilon}_{t-d} + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \overset{iid}{\sim} \mathcal{N}(0, \boldsymbol{Q}), \tag{4.6}$$

where $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_p$ are the $r \times r$ matrices of autoregressive coefficients for the factors and $\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_d$ are the $n \times n$ diagonal matrices of autoregressive coefficients for the idiosyncratic component (i.e. $\boldsymbol{\Phi}_1 = diag(\phi_1, \ldots, \phi_n)$). Specifically, the authors [93, 39] assume a vector autoregressive (VAR) process of order two ($p = 2$) for factors, and an autoregressive (AR) process of order one ($d = 1$) for the idiosyncratic components. Morevoer, $\boldsymbol{Q}$ is assumed to be a diagonal matrix, while restrictions on $\boldsymbol{U}$ can be imposed for identification purposes (i.e., $\boldsymbol{U} = \boldsymbol{I}_r$).

Such a structure can be constructed by adding to our formulation in equations (4.1) and (4.3) the following assumptions:

A.1 **Encoding function** $G_{\boldsymbol{\theta}_G}(\cdot) : \mathbf{y} \to \boldsymbol{f}$ is a linear operator;

A.2 **Decoding function** $F_{\boldsymbol{\theta}_F}(\cdot) : \boldsymbol{f} \to \tilde{\mathbf{y}}$ is a linear operator;

A.3 **Factor dynamics** $\boldsymbol{f}_t$ follows a linear stochastic vector difference equation;

A.4 **Idiosyncratic component dynamics** $\boldsymbol{\varepsilon}_t$ follows a linear stochastic vector difference equation with diagonal matrices of autoregressive coefficients;

A.5 **Distributions** Error terms from the transition (and emission) equations are assumed to be *i.i.d.* Gaussian.[9]

Autoencoders provide a practical solution for estimating factor models with a more general structure, by potentially relaxing one or more of these assumptions to obtain both nonlinear maps from reduced dimension factors to observable variables and vice-versa, but also to introduce nonlinear dynamic equations. This approach to the generalisation of dynamic factor models, is what we call Deep Dynamic Factor Models (D^2FMs or DDFM). In the next

[9] Once in state-space, a standard DFM as described in equations from (4.4) to (4.6) features a noise process in the measurement equation (4.4), on top of the error terms $\boldsymbol{u}_t$ and $\boldsymbol{\epsilon}_t$ from the transition equations. This measurement error term (call it $\boldsymbol{\eta}_t$) is usually assumed to be *i.i.d.* multivariate Gaussian with identity matrix scaled by a small constant, that is $\boldsymbol{\eta}_t \overset{iid}{\sim} \mathcal{N}(0, \eta \boldsymbol{I})$, with η a small number larger than zero.

section, we show how to construct and estimate an autoencoder that relaxes assumptions A.1 and A.2, while maintaining the others.[10]

4.2.3 Estimation and Conditional Likelihood

In principle the parameters of a parametric factor model of the form $\mathbf{y}_t = F_{\boldsymbol{\theta}_F}(\boldsymbol{f}_t) + \boldsymbol{\varepsilon}_t$ would be estimated via maximum likelihood,

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p_{model}(\boldsymbol{Y}|\widehat{\boldsymbol{Y}}) \ , \tag{4.7}$$

where by $\boldsymbol{Y}$ and $\widehat{\boldsymbol{Y}}$ we now indicate the full sample of observation and predicted values from the factor model, and $p_{model}(\cdot|\cdot)$ is the conditional probability density function of the model.

However, a direct maximum likelihood is rarely feasible, even for linear models, and iterative methods to find maximum likelihood or maximum a posteriori (MAP) estimates of the parameters of the model are preferred. In fact, maximum likelihood estimators of the parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_F, \boldsymbol{B}(L), \boldsymbol{U}, \boldsymbol{\Phi}(L), \boldsymbol{Q})$ are in general not available in closed form and a direct numerical maximisation can be too demanding when the cross-section is large. Indeed, a proposed solution in the linear factor model literature is to adopt the Expectation Maximisation (EM) algorithm, a maximum a posteriori method, and to initialise the common factors $\boldsymbol{f}_t$ with PCA on the observables.[11] The updates of the latent components are performed using the Kalman filter and smoother.

A similar approach can be selected from a DL point of view on factor models by employing the methodologies developed in the deep learning literature without losing the dynamic model interpretation. As we discuss in the next section, the model parameters of a D^{2}FM can be estimated via Markov Chain Monte Carlo gradient method, instead of using a least square based EM algorithm which is the standard practice for DFMs.

[10] See the dedicated background Section 3.1.1 and *Doz et al.* [154] for the robustness of the DFM described here to assumptions A.4 and A.5. With respect to the treatment of the time dimension into autoencoders using alternative approaches see Section 3.2.4.

[11] Estimation of linear factor models was originally carried out via simple principal component analysis (PCA).

It is well known, in the linear case, that if the innovation $\boldsymbol{\varepsilon}_t$ are assumed to be independent (or uncorrelated) of $\boldsymbol{f}_t$ and normally distributed, than the maximisation of the likelihood with respect to the parameters of the model yields the same estimate for the parameters as does minimising the mean squared error. Importantly, this equivalence between maximum likelihood estimation and minimisation of mean squared error holds regardless of the function used to predict the conditional mean of the conditionally Gaussian distributed variable $\mathbf{y}_t$ [see *Goodfellow et al.* 159, for example]. This allows for an interpretation of estimation results from autoencoders with mean squared error, from a Bayesian perspective using standard likelihood methods, or from a frequentist one as the (approximated) mean estimator of a Gaussian distributed process.

Furthermore, the equivalence between maximum likelihood estimation and minimisation of mean squared error together with the Universal Approximation Theorem, allows for the reinterpretation of autoencoders and the procedure adopted in estimating them as an efficient computational method for approximating the maximum likelihood estimates of nonlinear factor models. These are dynamic models defined by a conditionally Gaussian distribution centred around a mean provided by a nonlinear but continuous function of the inputs. In the next section, we provide an algorithm that implements these ideas.

4.3 D²FM Estimation

In this section we provide an algorithm to implement a Deep Dynamic Factor Model for macro and financial data. In its general form, the D²FM can be written as

$$\boldsymbol{f}_t = G_{\boldsymbol{\theta}_G}(\mathbf{y}_t) \ , \tag{4.8}$$

$$\mathbf{y}_t = F_{\boldsymbol{\theta}_F}(\boldsymbol{f}_t) + \boldsymbol{\varepsilon}_t = \widetilde{\mathbf{y}}_t + \boldsymbol{\varepsilon}_t \ , \tag{4.9}$$

$$\boldsymbol{f}_t = \boldsymbol{B}_1 \boldsymbol{f}_{t-1} + \cdots + \boldsymbol{B}_p \boldsymbol{f}_{t-p} + \boldsymbol{u}_t, \qquad \boldsymbol{u}_t \overset{iid}{\sim} \mathcal{N}(0, \boldsymbol{U}), \tag{4.10}$$

$$\boldsymbol{\varepsilon}_t = \boldsymbol{\Phi}_1 \boldsymbol{\varepsilon}_{t-1} + \cdots + \boldsymbol{\Phi}_d \boldsymbol{\varepsilon}_{t-d} + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \overset{iid}{\sim} \mathcal{N}(0, \boldsymbol{Q}), \tag{4.11}$$

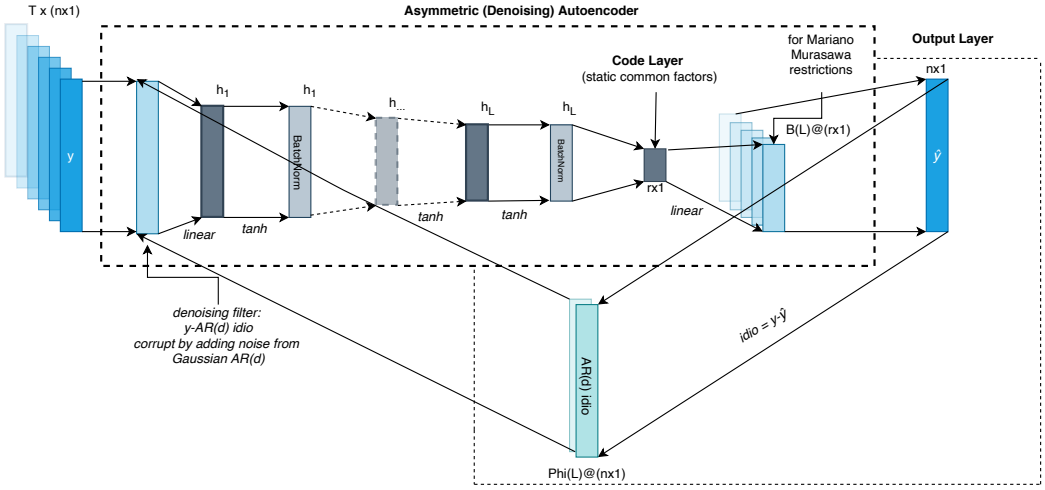


Figure 4.1: A graph representation of the training process for a the D^2FM with an asymmetric structure: nonlinear multilayer encoder and linear single layer decoder.

where $\boldsymbol{B}_i$ for $i = 1, \dots, p$ and $\boldsymbol{U}$ are left unrestricted, while $\boldsymbol{\Phi}_i$ for $i = 1, \dots, d$ and $\boldsymbol{Q}$ are assumed to be diagonal. The assumptions on the linearity of the dynamic equations are maintained (equations (4.10) and (4.11)), while the model allows for a nonlinear map between variables and factors. The estimation of linear factor dynamics separately yields to what *Stock and Watson* [149] call a *state space with static (common) factors*, as opposed to a *state space with dynamic (common) factors*.[12]

The D^2FM can be implemented using a symmetric autoencoder structure with an MLP capturing the encoding function (in equation (4.8)), and another MLP providing the decoding one (in equation (4.9)). The assumption of *i.i.d.* and Gaussian innovations allows for an interpretation of the estimated network when minimising the mean squared errors loss as MAP of the likelihood of the model [159].

Importantly, such a model specification encompasses several simplified models, most notably the standard linear DFMs, and hence the estimation algorithm can be specialised to the scope.

4.3.1 Network Design

The core of the model is provided by an autoencoder with a nonlinear multi-layer encoder, and either a symmetric structure in the decoding (for nonlinear decoding) or an asymmetric structure with a linear single layer decoder. Linear stochastic autoregressive equations are chosen to model the dynamics of factors and idiosyncratic components. Alternatively, one could employ nonlinear dynamics in the form of multi-layer perceptrons (MLPs) or Long Short Term Memory (LSTM). Figure 4.1 shows a diagrammatic representation of the model.

The number of hidden layers in the encoding network, in addition to the number of neurons need not be pre-specified but can be selected via cross-validation. With respect to the preferred activation functions, we equip each neuron in the coding layers with a link function in the form of the hyperbolic tangent (*tanh*) for the real-time macroeconomic dataset, and of the rectified linear unit (*relu*) for the Monte Carlo exercises.[13] In the encoding multilayer perceptron we also include two batch normalisation layers to allow for some regularisation and control over potential *covariate shift* [268].[14]

In the decoding network, an additional linear layer can be included to introduce constraints needed to account for the mixed frequencies of macroeconomic data. This additional layer does not have any additional parameter, and it only includes restrictions on the output data structure. Finally, we allow for idiosyncratic biases by including a bias term into the output layer of the network.

4.3.2 Estimation and Online Learning of the D^2FM

In our estimation of the D^2FM, we propose a two-step procedure to differentiate between on-line and off-line learning.

- **Step 1** estimate off-line all the parameters of the model;

[12] See Section 3.1 for a more detailed discussion about different types of DFMs.

[13] In general, some tuning is needed for the user in order to find the correct specification for the dataset at use.

[14] These two normalisation layers can also improve the stability of the gradient updates among batches and potentially make the optimisation smoother, hence allowing for faster training [269].

- **Step 2** cast the decoding part in a state-space framework to allow for on-line updates of the latent states given the observables.

Algorithm 2 implements the off-line estimation step (**Step 1**) of our D²FM, assuming an AR(d) for $p_{idio}(.)$, but possibly a general encoding $G_{\boldsymbol{\theta}_G}(\cdot)$ and decoding $F_{\boldsymbol{\theta}_F}(\cdot)$ function. The proposed algorithm for estimating D²FM builds on and extends what has been proposed by *Bengio et al.* [194] to estimate Generalised Denoising Autoencoders.

Algorithm 2 MCMC for D²FM with stationary AR(d) idiosyncratic components – *requires a training set, an encoding structure* $G_{\boldsymbol{\theta}_G}(\cdot)$ *and a decoding one* $F_{\boldsymbol{\theta}_F}(\cdot)$

init: $\boldsymbol{\theta}_G, \boldsymbol{\theta}_F, \boldsymbol{\Phi}, \Sigma_\varepsilon, \boldsymbol{\varepsilon}_t$
repeat
1: $\widetilde{\mathbf{y}}_t|(\mathbf{y}_t, \hat{\boldsymbol{\varepsilon}}_t) = \mathbf{y}_t - \boldsymbol{\Phi}(L)\boldsymbol{\varepsilon}_t$
2: **Loop** epochs, batches **Do**
3: draw $\boldsymbol{\varepsilon}_t^{(mc)} \overset{iid}{\sim} \mathcal{N}(0, \Sigma_\varepsilon)$
4: $\mathbf{y}_t^{(mc)} = \widetilde{\mathbf{y}}_t|(\mathbf{y}_t, \hat{\boldsymbol{\varepsilon}}_t) + \boldsymbol{\varepsilon}_t^{(mc)}$
5: θ_G, θ_F update by a gradient based step on $\hat{\mathcal{L}}(\mathbf{y}_t, F_{\boldsymbol{\theta}_F}(G_{\boldsymbol{\theta}_G}(\mathbf{y}_t^{(mc)})))$
6: **End Loop**
7: $\boldsymbol{f}_t|\mathbf{y}_t^{(mc)} = \mathbb{E}_{\mathbf{y}_t^{(mc)} \sim \mathbf{y}_t, \hat{\boldsymbol{\varepsilon}}_t} G_{\boldsymbol{\theta}_G}(\mathbf{y}_t^{(mc)})$
8: $\boldsymbol{\varepsilon}_t|\mathbf{y}_t, \boldsymbol{f}_t = \mathbf{y}_t - F_{\boldsymbol{\theta}_F}(\boldsymbol{f}_t|\mathbf{y}_t^{(mc)})$
9: $\boldsymbol{\Phi} \leftarrow stationary\ AR(d)$ on $\boldsymbol{\varepsilon}_t$
10: $\Sigma_\varepsilon \leftarrow from\ \boldsymbol{\varepsilon}_t$
until convergence on $\hat{\mathcal{L}}(\mathbf{y}_t, F_{\boldsymbol{\theta}_F}(\boldsymbol{f}_t|\mathbf{y}_t^{(mc)}))$ in L_1 norm
return $\Sigma_\varepsilon, \boldsymbol{\Phi}, \boldsymbol{f}_t, F_{\boldsymbol{\theta}_F}$

Let us summarise the estimation algorithm. Parameters are first initialised. Line 1 performs a filtering of the input data $\mathbf{y}_t$ by using the conditional mean of the AR(d) of the idiosyncratic components. From lines 2 to 6, the Monte Carlo step and the gradient updates over each epoch and batch are carried out, employing the filtered data $\widetilde{\mathbf{y}}_t$ and injecting Gaussian noise from $\boldsymbol{\varepsilon}_t$ in a denoising fashion to obtain the noisy observations $\mathbf{y}_t^{(mc)}$. In line 7, the latent states $\boldsymbol{f}_t$ are extracted from the encoding network via Monte Carlo integration, while from line 8 to 10 the algorithm updates the parameters of the idiosyncratic

process $\boldsymbol{\varepsilon}_t$, conditional on the factors and the observables. The adoption of an L_2 (MSE) loss function $\hat{\mathcal{L}}(\mathbf{y}_t, F_{\boldsymbol{\theta}_F}(\boldsymbol{f}_t|\mathbf{y}_t^{(mc)}))$ allows for interpretability of the results, as discussed. We specify an estimated loss, as missing data prevents us from deriving the exact loss.[15] Finally, convergence is checked as the L_1 norm of the distance between the loss function at two iterations. It is worth noting that the loss, $\hat{\mathcal{L}}(\cdot)$ includes only the common components, since under our assumptions at convergence we have the following decomposition of the log-likehood:

$$\begin{aligned} &\log p_{model}(\mathbf{y}_t|\boldsymbol{f}_t = G_{\boldsymbol{\theta}_G}(\mathbf{y}_t^{(mc)}), \boldsymbol{\varepsilon}_t = \hat{\boldsymbol{\varepsilon}}_t) = \\ &\log p_{decoder}(\mathbf{y}_t|\boldsymbol{f}_t = G_{\boldsymbol{\theta}_G}(\mathbf{y}_t^{(mc)})) + \log p_{idio}(\mathbf{y}_t|\boldsymbol{\varepsilon}_t = \hat{\boldsymbol{\varepsilon}}_t) \ , \end{aligned} \tag{4.12}$$

where $\hat{\boldsymbol{\varepsilon}}_t$ is the estimated idiosyncratic autoregressive component. In running over epochs and batches (lines 2 to 6), the algorithm injects uncorrelated noise into the data (it is a Denoising Autoencoder). Hence it searches for a maximum a posteriori (MAP) of the parameters for the modified model with log-likelihood

$$\mathbb{E}_{\mathbf{y}_t \sim\ p_{data}(\mathbf{y}_t)} \mathbb{E}_{\mathbf{y}_t^{(mc)} \sim p_{noisy}(\mathbf{y}_t^{(mc)}|\mathbf{y}_t, \hat{\boldsymbol{\varepsilon}}_t)} \log p_{model}(\mathbf{y}_t|\boldsymbol{f}_t = G_{\boldsymbol{\theta}_G}(\mathbf{y}_t^{(mc)}), \boldsymbol{\varepsilon}_t = \hat{\boldsymbol{\varepsilon}}_t) \ , \tag{4.13}$$

where $p_{noisy}(\mathbf{y}_t^{(mc)}|\mathbf{y}_t, \hat{\boldsymbol{\varepsilon}}_t)$ is the corruption distribution, using a Gaussian autoregressive process. The idea behind this procedure is to filter out the foreseeable idiosyncratic part from the input variables, so that only the common component(s) remain(s). Injecting noise from the unconditional idiosyncratic distribution will generate new samples which are not unreasonably far from the old ones. In doing so, we define an appealing and convenient linkage between the corruption process of the denoising approach and the idiosyncratic component distribution (p_{idio}).

In **Step 2**, the output of the algorithm is cast in the state-space of equations (4.9)-(4.11). Dynamics of the common factors are estimated via OLS or Maximum Likelihood.[16] State updates can then be carried out via either

[15] We give details about the treatment of missing data in Section 4.4.1.

[16] The dynamic of the common latent states can also be estimated directly in Algorithm 2.

nonlinear filtering procedures for a nonlinear decoder, or via Kalman filtering in the presence of a linear decoder. This allows for online (i.e., out-of-sample) learning with the flow of data.

4.3.3 Hyperparameter Selection

The D^2FM described is subject to the selection of a number of critical parameters determining its structure beyond $\boldsymbol{\theta}$. These parameters are commonly known as hyperparameters, in that they are set before the training starts and usually selected over a grid with respect to some validation loss, which is generally estimated via a process called cross-validation.

The D^2FM has hyperparameters typical of both deep learning and time-series models. In particular, the deep learning hyperparameters can be divided into two categories. The first relates to the neural network structure and includes: type of layers, number of hidden layers, number of neurons per each hidden layer, penalisation coefficients, dropout layers and relative dropout rates (if included), batch normalisation layers and the link function used. The second category relates to the optimisation algorithm used and comprehends: size of the mini-batches, number of epochs, the learning rate and the momentum coefficients of the gradient optimisation method, if present. Standard time-series factor models have a few additional hyperparameters which include: number of latent common states, number of lags of the input variables, number of lags of the latent common states and of the idiosyncratic states. These hyperparameters, in the time series literature, are either fixed a-priori or estimated using information criteria instead of using cross validation methods.[17]

It is important to mention that in time-series we cannot apply the common *K-fold* cross-validation method because of possible serial data correlation in the residuals, unless this is absent and some other conditions are met [115].

This can be achieved by explicitly including lagged values of the factors before the decoding network ($F_{\boldsymbol{\theta}_F}$ of the algorithm), or via the use of recurrent layers.

[17]The Akaike information criteria (AIC) and the Bayesian information criteria (BIC) can be used to determine the number of lags, while the number of latent factors can, in principle, be estimated using the method proposed by *Alessi et al.* [270], which improves the *Bai and Ng* [152] methodology.

Therefore, we use a standard out-of-sample validation approach consisting of splitting the available set of observations up to a certain point in time, T, between a training set $[0, T-k*h-1]$, and a validation set $[T-k*h, T-(k-1)*h]$, where h determines the length of the set, and $k = K, \ldots, 1$ with $K \ll \frac{T-1}{h}$. By averaging over the losses computed on the K *validation sets*, we get an estimate of the *validation loss* which is consistent when conditions in *Bergmeir et al.* [115] are met. Notably, we need to estimate a given model with fixed hyperparameter K times, and this for each possible combination of hyperparameters. Thus, with a grid search method the computational cost is exponential in the number of hyperparameters. Indeed, alternative methods based on stochastic search are available [see 271, for example], as well as other methods based on evolutionary algorithms [272].

4.4 A Deep Dynamic Factor Model for Macroeconomic and Financial Data

We estimate a simplified version of the D^{2}FM in the empirical application with a linear mapping between the factors and the variables (see Figure 4.1), i.e.,

$$\mathbf{y}_t = \boldsymbol{\theta}_F \boldsymbol{f}_t + \boldsymbol{\varepsilon}_t \ . \tag{4.9'}$$

It is worth mentioning, that in this form, the model can be seen as a very flexible generalisation of the approach of *Bai and Ng* [264] who propose to extract factors from variables as well as their squared values and their crossproducts.

There are a few advantages to considering this simpler D^{2}FM. First, the model maintains the same level of interpretability as a standard DFM, hence making it easy to compare the two models. Certainly, this simple architecture is inspired by the recent work of *Rudin* [28] that has compelled the design of inherently interpretable models, as opposed to a purely *black box* approach. Second, while interpretable, the autoencoder structure allows us to introduce deep learning techniques in this framework to test its potential towards the

construction of more general models. Third, the linear decoding network and the linear state-space framework allow us to employ a standard Kalman Filter to update the unobservable states in real-time. Finally, the adoption of linear filtering techniques, in turn, allows for an efficient computation and easy interpretation of the model forecast revisions coming from the flow of data onto the performances of the model, as in *Banbura et al.* [93].

4.4.1 Mixed Frequency and Missing Data

Economic data are rarely available all at the same frequency – be it weekly, monthly or quarterly – and missing data are a feature of real-time macroeconomic datasets, that are characterised by the non-synchronous and staggered data release of new data points from statistical offices. The model accounts for these two features of macro data.

We handle the missing data problem in two or three steps depending on the dataset. Namely, if in the pre-training when dropping missing values we are left with few observations,[18] then we first initialise missing values with spline. Otherwise this first step is omitted and the pre-training is carried out only on non missing data points. Second, we iterate the parameters maximisation by replacing the missing data in the full sample with fitted values obtained by conditioning on the estimated model and on the realisation of the latent factors. Maximisation is carried out only on non-missing points through the use of mask matrices; the number of observations over which the gradients are computed can therefore differ across dimensions. Finally, in the real-time online update phase (out-of-sample), we employ the Kalman filter to accommodate for missing data [see 93, 39, 259, for example].

In dealing with mixed frequency data, several options are possible [see 273, 274, for example]. When the dataset includes monthly and quarterly variables, the most popular option is the *Mariano and Murasawa* [252] approximation. In the model, this approximation is implemented by including an additional final layer to the decoding network to allow for monthly aggregations to the

[18] In the empirical section we set this to 50.

quarterly variables. This layer has fixed weights which are not subject to the optimisation. In particular, and taking the example of a quarterly growth rate (y_t^q), it is possible to link it to the unobserved monthly growth rate (y_t^m) in the previous layer, where every third observation of y_t^q is given by:

$$y_t^q = \frac{1}{3}y_t^m + \frac{2}{3}y_{t-1}^m + y_{t-2}^m + \frac{2}{3}y_{t-3}^m + \frac{1}{3}y_{t-4}^m. \tag{4.14}$$

4.4.2 Model Specification and Training Details

The core of the model is provided by an asymmetric autoencoder with a nonlinear multilayer encoder and a linear single layer decoding structure. Table 4.1 provides a summary of the network design choices, and reports the choices operated for each hyperparameter of our model; a number of these are selected via out-of-sample validation (see 4.3.3).

Model Components		Hyperparameter	Choice taken
Autoencoder	**Model Structure**	number of hidden layers	3
		number of neurons for each layer	selected via cross-validation
		penalisation	none
		dropout layers and rates	none
		batch norm layers	2 included in the encoding network
		link function	tanh or relu
	Optimization	size of mini batches	100 monthly observations
		number of epochs	100 for each MC iteration
		optimisation algorithm	ADAM with default parameters
Dynamic Equations	**Model Structure**	number of latent states	selected via cross-validation
		number of lags input variables	selected via cross-validation
		number of lags for latent common states	2 as in *Banbura and Modugno* [39]
		number of lags for idiosyncratic states	1 as in *Banbura and Modugno* [39]

Table 4.1: Summary of model features and choices.

Optimisation, both during pre-training and training is carried out by using ADAM [167] with default hyperparameters and 100 epochs. Before starting the training, ADAM is reinitialised and then is run on batches (i.e., subsamples) with a size of at least 100 monthly observations (approximately 8 years, the average duration of a business cycle). In the training phase we set again the number of epochs (runs on the full sample) to 100 for each iteration of the MCMC. These iterations are also used to update the idiosyncratic distribution.

Parameters are initialised in a two stage approach in our empirical model. First, by using Xavier initialisation – weights in the link functions are sampled from a Gaussian distribution with zero mean and a variance of $2/(n_{in}+n_{out})$, where n_{in} is the number of input units and n_{out} is the number of output units, [see 275], and second by performing a pre-training step using a standard autoencoder on a full dataset where the rows containing missing data are discarded.[19] This pre-training procedure is needed to warm up the chain.

4.5 Monte Carlo Experiment

In this section we carry out a Monte Carlo experiment to compare on a known data generating process (DGP) the performances of the DFM and its more general version: the D^2FM. The experiments combine the simulation environments of *Doz et al.* [154], *Banbura and Modugno* [39], and *Gu et al.* [22], the latter to extend the formers' observable equation to nonlinearities.

4.5.1 Experimental Set-up

We assume the following data generating process (DGP) [154, 39]:

$$\mathbf{y}_t = F(\boldsymbol{f}_t) + \boldsymbol{\varepsilon}_t \ , \tag{4.15}$$

$$\boldsymbol{f}_t = \boldsymbol{B}_1 \boldsymbol{f}_{t-1} + \boldsymbol{u}_t, \qquad \boldsymbol{u}_t \overset{iid}{\sim} \mathcal{N}(0, \boldsymbol{I}_r), \tag{4.16}$$

$$\boldsymbol{\varepsilon}_t = \boldsymbol{\Phi}_1 \boldsymbol{\varepsilon}_{t-1} + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \overset{iid}{\sim} \mathcal{N}(0, \boldsymbol{Q}), \tag{4.17}$$

with $t = 1, \ldots, T$. However, to include nonlinearities in the DGP, we specify factor loadings similar to *Gu et al.* [22] and allow $F(.)$ to take two forms:

$$F(\boldsymbol{f}_t) = \begin{cases} \boldsymbol{\Lambda} \boldsymbol{f}_t & \text{if linear} \\ \boldsymbol{\Lambda}[\boldsymbol{f}_t, poly(\boldsymbol{f}_t, 2), sgn(\boldsymbol{f}_t)]' & \text{if nonlinear} \end{cases} \tag{4.18}$$

[19] In particular, in the empirical application we check that at least 50 observations are present when applying this rule. If this is not the case, then we drop observations for which the corresponding variables are missing for more than 20% of the total number of features, and we fill the rest with splines (see Section 4.4.1).

where $sgn()$ is the sign function (1 if positive, -1 if negative) and $poly(.,2)$ is a polynomial function generator of order 2; while, all other parameters are as in *Banbura and Modugno* [39],

$$\mathbf{\Lambda}_{ij} \overset{iid}{\sim} \mathcal{N}(0,1),\ i=1,\dots,n, j=1,\dots,\tilde{r}, \tilde{r} = \begin{cases} r & \text{linear} \\ \frac{4r+r(r+1)}{2} & \text{nonlinear} \end{cases} \tag{4.19}$$

$$\boldsymbol{B}_{ij,1} = \begin{cases} \rho & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}, \qquad \boldsymbol{\Phi}_{ij,1} = \begin{cases} \alpha & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}, \tag{4.20}$$

$$\boldsymbol{Q}_{ij} = \tau^{|i-j|}(1-\alpha^2)\sqrt{\gamma_i \gamma_j}, \quad \gamma_i = \frac{\beta_i}{1-\beta_i}\frac{1}{1-\rho^2}\sum_{j=1}^{\tilde{r}} \mathbf{\Lambda}_{ij}, \quad \beta_i \sim U([u,1-u]). \tag{4.21}$$

To balance the computational cost with the generality of the experimental framework, we restrict the range of possible configurations for the free parameters to the following: $r=\{1,3\}$, $n=\{10,100\}$, $\rho=\{0.5,0.9\}$, $\alpha=\{0,0.5\}$, $T=\{50,200\}$ and the fraction of missings is in $\{0,0.3\}$. For each setting, we run 100 Monte Carlo simulations and estimate a DFM, and a four layers D^{2}FM with *relu* nonlinearities augumented with three *BatchNorm* layers. The number of factors is set for both models to the true number of factors (i.e., r when the DGP is linear in the factors and $\tilde{r}$ when it is nonlinear). For the D^{2}FM, and starting from the factor layer, hidden neurons increase by a factor of two in each layer up to the input layer.

We compare the models based on the trace R^2 of the regression of the estimated factors on the true ones [85, 154, 39]:

$$\frac{Trace(\boldsymbol{F}'\hat{\boldsymbol{F}}(\hat{\boldsymbol{F}}'\hat{\boldsymbol{F}})^{-1}\hat{\boldsymbol{F}}'\boldsymbol{F})}{Trace(\boldsymbol{F}'\boldsymbol{F})} \tag{4.22}$$

where $\hat{\boldsymbol{F}} = \mathbb{E}_{\hat{\boldsymbol{\theta}}}[\boldsymbol{F}|H_T]$ and H_T is the history of the data, while $\boldsymbol{F}$ represent the whole history of the true factors, including their nonlinear transformations in the nonlinear case.

4.5.2 Results and Discussion

In Table 4.2 we report results for the linear case. Differences in performances between the D^{2}FM and the DFM, although often significant, are very small and go in both directions depending on the specific case. Thus, signalling the two frameworks are generally equivalent when the data generating process (DGP) is linear.

Table 4.3 shows results for the nonlinear case; here the difference in performance is striking and in favour of the D^{2}FM. In particular, all the differences are statistically significant and the D^{2}FM estimates can explain between 15% and 34% more of the total variance of the true factors compared to the DFM.

4.6 Encoding the US Economy in Real Time

In this section we report the empirical results of the model presented in Section 4.4. Specifically, the performances of the model are tested in forecasting, nowcasting and backcasting using a fully real-time big US macro dataset, and assessing against three benchmark models:[20] (i) a univariate AR(1) statistical benchmark; (ii) a state-of-the-art DFM with two and (iii) three latent factors, estimated via quasi maximum likelihood as proposed by *Giannone et al.* [255] and generalised in *Banbura and Modugno* [39] (we refer to this model as DFM-EM). The model is multitarget but we mainly focus on US GDP. This exercise can be seen as a validation test to check whether the model is able to correctly capture the relevant features of the data generating process on a real case too.

4.6.1 A Real-Time *Big* Macro Dataset

To test its capability, we estimate the model by encoding a real-time version of the full *McCracken and Ng* [250] FRED-MD dataset, a large macroeconomic database for the U.S. economy, specifically designed for the empirical analysis of

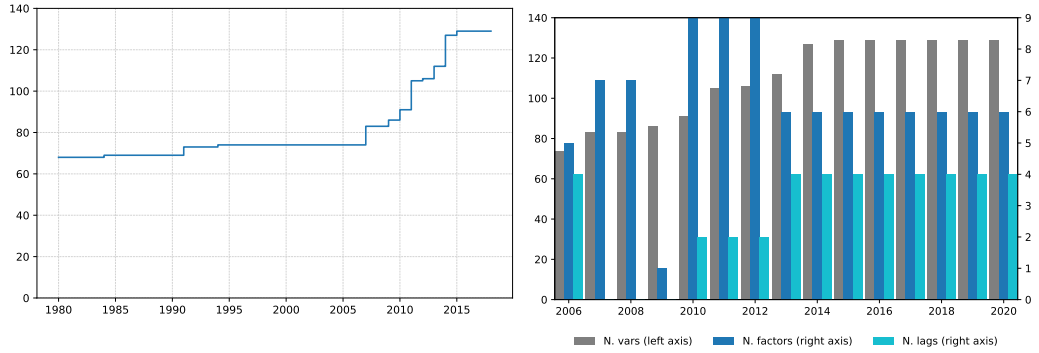
[20] Backcast is the estimate of the previous quarter up to the official release date; nowcast is the estimate of the current quarter up to the official release date, and forecast is the estimate of the next quarter up to the the official release date. We are able to produce backcast values because the GDP is released usually 5 weeks after the end of the reference quarter, hence we use the releases of the other variables during these 5 weeks to update the backcast figure.

Factors				1					
Sample				50			200		
α	ρ	N vars	Missings	D^2FM	DFM	Diff.	D^2FM	DFM	Diff.
0	0.5	10	0	0.91	0.89	**0.025*** **	0.94	0.94	-0.008***
0	0.5	10	0.3	0.88	0.83	**0.045*** **	0.91	0.91	-0.006***
0	0.5	100	0	0.96	0.95	**0.011*** **	0.99	0.99	-0.001***
0	0.5	100	0.3	0.95	0.93	**0.02*** **	0.99	0.99	**0.001**
0	0.9	10	0	0.74	0.75	-0.011	0.94	0.95	-0.01***
0	0.9	10	0.3	0.71	0.70	**0.001* **	0.93	0.94	-0.013***
0	0.9	100	0	0.77	0.74	**0.024**	0.96	0.96	**0.001*** **
0	0.9	100	0.3	0.76	0.75	**0.017*** **	0.96	0.96	**0.002**
0.5	0.5	10	0	0.90	0.85	**0.043*** **	0.92	0.92	**0.001**
0.5	0.5	10	0.3	0.85	0.77	**0.086*** **	0.88	0.89	-0.008
0.5	0.5	100	0	0.96	0.94	**0.013*** **	0.99	0.99	-0.001***
0.5	0.5	100	0.3	0.95	0.94	**0.015*** **	0.98	0.99	-0.001***
0.5	0.9	10	0	0.72	0.72	**0.004*** **	0.93	0.93	0
0.5	0.9	10	0.3	0.71	0.70	**0.007*** **	0.92	0.93	-0.004***
0.5	0.9	100	0	0.77	0.73	**0.035** **	0.96	0.96	**0.001*** **
0.5	0.9	100	0.3	0.76	0.75	**0.018*** **	0.96	0.96	**0.002*** **
Factors				3					
Sample				50			200		
α	ρ	N vars	Missings	D^2FM	DFM	Diff.	D^2FM	DFM	Diff.
0	0.5	10	0	0.71	0.71	-0.001**	0.76	0.80	-0.039***
0	0.5	10	0.3	0.60	0.58	**0.021**	0.66	0.71	-0.051***
0	0.5	100	0	0.94	0.91	**0.021*** **	0.97	0.97	-0.002***
0	0.5	100	0.3	0.92	0.91	**0.014*** **	0.96	0.96	**0.002**
0	0.9	10	0	0.63	0.66	-0.029***	0.82	0.88	-0.06***
0	0.9	10	0.3	0.58	0.64	-0.066***	0.75	0.85	-0.101***
0	0.9	100	0	0.74	0.74	**0.003*** **	0.92	0.92	-0.001***
0	0.9	100	0.3	0.73	0.73	-0.002	0.92	0.92	-0.003***
0.5	0.5	10	0	0.67	0.63	**0.044*** **	0.70	0.69	**0.01**
0.5	0.5	10	0.3	0.56	0.52	**0.035*** **	0.60	0.61	-0.013***
0.5	0.5	100	0	0.93	0.91	**0.021*** **	0.97	0.97	0
0.5	0.5	100	0.3	0.92	0.88	**0.033*** **	0.96	0.95	**0.002**
0.5	0.9	10	0	0.60	0.63	-0.031***	0.77	0.85	-0.083***
0.5	0.9	10	0.3	0.55	0.61	-0.063***	0.70	0.82	-0.12***
0.5	0.9	100	0	0.74	0.74	**0.001*** **	0.92	0.92	0
0.5	0.9	100	0.3	0.73	0.72	**0.01** **	0.92	0.92	-0.002***

Table 4.2: Linear DGP. Median over 100 Monte Carlo simulations of the Trace of the R^2 between estimated and true factors. The difference is computed as: $R^2_{D^2FM} - R^2_{DFM}$. Significance levels are based on a two sided Wilcoxon signed-rank test: * for 10%, ** for 5% and *** for 1%.

Factors				1					
Sample				50			200		
α	ρ	N vars	Missings	D^2FM	DFM	Diff.	D^2FM	DFM	Diff.
0	0.5	10	0	0.849	0.63	**0.223*** **	0.88	0.66	**0.217*** **
0	0.5	10	0.3	0.791	0.55	**0.245*** **	0.827	0.61	**0.22*** **
0	0.5	100	0	0.908	0.72	**0.187*** **	0.911	0.76	**0.154*** **
0	0.5	100	0.3	0.906	0.7	**0.208*** **	0.912	0.74	**0.17*** **
0	0.9	10	0	0.93	0.6	**0.335*** **	0.945	0.64	**0.302*** **
0	0.9	10	0.3	0.914	0.59	**0.323*** **	0.94	0.64	**0.299*** **
0	0.9	100	0	0.941	0.61	**0.334*** **	0.96	0.65	**0.308*** **
0	0.9	100	0.3	0.947	0.61	**0.336*** **	0.962	0.66	**0.305*** **
0.5	0.5	10	0	0.862	0.59	**0.274*** **	0.87	0.63	**0.241*** **
0.5	0.5	10	0.3	0.787	0.51	**0.276*** **	0.802	0.58	**0.222*** **
0.5	0.5	100	0	0.913	0.71	**0.199*** **	0.912	0.75	**0.161*** **
0.5	0.5	100	0.3	0.908	0.69	**0.216*** **	0.911	0.74	**0.17*** **
0.5	0.9	10	0	0.932	0.59	**0.342*** **	0.948	0.64	**0.308*** **
0.5	0.9	10	0.3	0.909	0.6	**0.314*** **	0.934	0.64	**0.295*** **
0.5	0.9	100	0	0.929	0.61	**0.322*** **	0.962	0.65	**0.31*** **
0.5	0.9	100	0.3	0.941	0.61	**0.332*** **	0.961	0.66	**0.303*** **
Factors				3					
Sample				50			200		
α	ρ	N vars	Missings	D^2FM	DFM	Diff.	D^2FM	DFM	Diff.
0	0.5	10	0	0.741	0.51	**0.228*** **	0.662	0.44	**0.224*** **
0	0.5	10	0.3	0.653	0.43	**0.223*** **	0.547	0.34	**0.203*** **
0	0.5	100	0	0.927	0.74	**0.191*** **	0.948	0.76	**0.184*** **
0	0.5	100	0.3	0.871	0.68	**0.188*** **	0.924	0.73	**0.193*** **
0	0.9	10	0	0.94	0.61	**0.332*** **	0.926	0.65	**0.274*** **
0	0.9	10	0.3	0.884	0.61	**0.279*** **	0.863	0.64	**0.226*** **
0	0.9	100	0	0.978	0.67	**0.309*** **	0.987	0.73	**0.253*** **
0	0.9	100	0.3	0.974	0.68	**0.296*** **	0.984	0.75	**0.232*** **
0.5	0.5	10	0	0.735	0.51	**0.227*** **	0.638	0.42	**0.222*** **
0.5	0.5	10	0.3	0.646	0.43	**0.213*** **	0.526	0.34	**0.189*** **
0.5	0.5	100	0	0.907	0.72	**0.189*** **	0.936	0.75	**0.188*** **
0.5	0.5	100	0.3	0.85	0.66	**0.191*** **	0.91	0.71	**0.197*** **
0.5	0.9	10	0	0.942	0.61	**0.332*** **	0.922	0.64	**0.278*** **
0.5	0.9	10	0.3	0.884	0.62	**0.267*** **	0.857	0.64	**0.222*** **
0.5	0.9	100	0	0.978	0.67	**0.31*** **	0.985	0.73	**0.253*** **
0.5	0.9	100	0.3	0.974	0.68	**0.298*** **	0.982	0.74	**0.238*** **

Table 4.3: Nonlinear DGP. Median over 100 Monte Carlo simulations of the Trace of the R^2 between estimated and true factors. The difference is computed as: $R^2_{D^2FM} - R^2_{DFM}$. Significance levels are based on a two sided Wilcoxon signed-rank test: * for 10%, ** for 5% and *** for 1%.



(a) Number of variables. **(b)** Number of variables, factors and lags.

Figure 4.2: Panel (a) reports the number of variables along the entire considered time. Panel (b) reports the number of variables, factors and lags selected over time via out-of-sample validation as described in Section 4.3.3. The grey bars represent the number of variables available for each year (left axis); blue bars represent the optimal number of latent common states (right axis); and cyan bars represent the optimal number of lags of input variables (right axis). The x-axis shows the year during which the model is used for the out-of-sample evaluation.

big data.[21] The cross section of data is mixed frequency because it includes 128 monthly indicators and Real GDP, which is a quarterly indicator. All the data are stationarised and standardised following the specifications in *McCracken and Ng* [250].[22] In Tables A.1-A.4, we also report the respective publication delay (in days) of each series. There are substantial differences in the timeliness of different variables. Some are more timely (e.g., *soft* indicators or surveys), while others are released with one to two months of delay (usually *hard* data on real activity).

The vintages in the dataset span the period January 1980 to May 2020. Figure 4.2a reports the number of variables available across time periods. We first estimate the model using the data up to December 2005, and then we perform an expanding window forecasting exercise starting form the 1st of January 2006, hence our test sample goes from 1st of January 2006 to 31st of May 2020, including the Great Recession in 2007-2009. A data vintage is

[21] We marginally extend the dataset by including two Purchasing Managers' Indices (PMIs), since they are considered to be important indicators for nowcasting and do not get revised over time.

[22] In the Appendix Tables A.1-A.4 provide the complete list of the variables used and their transformation codes.

created every time a new time-series data point is released, and it contains all the data available up to that point in time, including also data revisions. The real-time infrastructure adapts automatically to the expanding number of variables used as input for the model. For each iteration, as new data arrive, the model is re-evaluated and outputs a sequence of backcasts-nowcasts-forecasts for GDP and all the other variables. These forecasts are conditional only to the real-time information set, i.e. only data available up to that specific point in time without taking into consideration further revisions.

Many of the hyperparameters determining the model specification are not fixed ex-ante but are instead selected in an intensive out-of-sample validation exercise, as reported in Table 4.1, and discussed in Section 4.3.3. The real-time validation exercise also provides information on the ability of the model to change its optimal hyperparameter specification over time as new data comes in (the validation length is set to one year). Figure 4.2b shows the evolution of the number of factors and lags that are selected over the sample via the validation method described in Section 4.3.3.

4.6.2 Model Evaluation

Figure 4.3 shows the nowcast reconstruction in real-time for the D^2FM, the DFM-EM with 2 and 3 factors, and the AR(1) model against the realised quarterly U.S. GDP. Overall, the D^2FM and the DFM-EM models provide a similar assessment of the state of the economy in the nowcasting horizon, although the D^2FM is slightly more accurate.[23]

We formally assess the performance(s) of the model – and of the AR(1), the DFM-EM with 2 and 3 factors – by computing root mean square forecast error (RMSFE). This metric is updated every time the data vintage gets updated due to a new data release. We report both an overall RMSFE (Table 4.4) that gives us a synthetic value about the performance of each model on the entire

[23]In particular, differences in predictions between the D^2FM and its competitors become statistically significant at least at the 10% level starting from 31 days after the beginning of the reference quarter, as of the Diebold-Mariano test; whilst, they are not statistically different for the remaining part of the nowcasting horizon and during the backcast period.

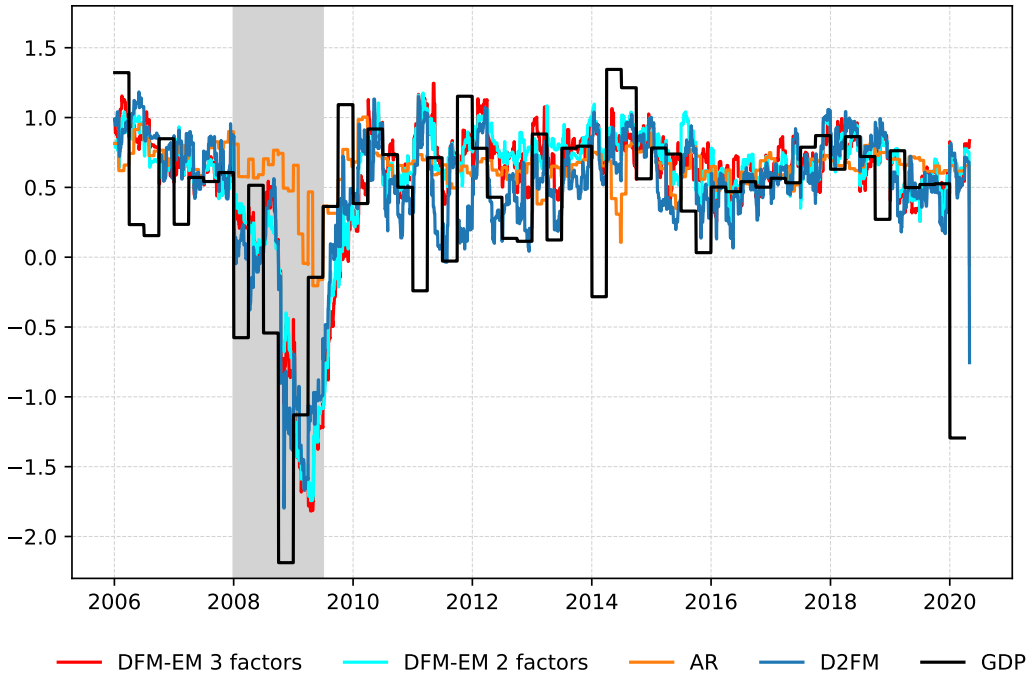


Figure 4.3: This Figure shows the nowcast reconstruction in real-time of the D^{2}FM, DFM-EM with 2 and 3 factors and the AR(1) versus the growth rate of the U.S. GDP. Shaded area is the NBER recession period.

out of sample set, and a dynamic RMSFE (Figure 4.4) that illustrates how the RMSFE evolves from the forecast period to the backcast period, until the day before the release. Results indicate that the D^{2}FM is able to outperform all the competitors during the entire forecast and nowcast period. The gain in terms of performance achieved by the D^{2}FM in these two periods is quite considerable, and reflects this model's ability to better compress the useful information reducing the level of uncertainty.

The model also delivers forecasts for all other variables. Table 4.5 reports the average of the RMSFEs of the D^{2}FM over all of the monthly variables, in ratio to the AR(1) RMSFEs. The D^{2}FM beats the AR(1) over all the horizons – the backcast improves by 10%; the nowcast improves by 20%; and the forecast improves by 18%.[24]

[24]Overall, the D^{2}FM improves the prediction accuracy for roughly 80% of the monthly variables included in the dataset with respect to the AR(1).

	Forecasting		Nowcasting			Backcasting
Model	30 weeks	26 weeks	20 weeks	14 weeks	8 weeks	2 weeks
D^2FM	**0.895**	**0.906**	**0.895**	**0.798**	**0.839**	0.832
DFM-EM 3	1.032	1.034	0.973	0.87	0.869	**0.826**
DFM-EM 2	1.015	1.027	0.962	0.894	0.886	0.858

Table 4.4: Comparison of RMSEs relative to the AR(1) benchmark

Notes: This table reports the RMSE of the D^2FM, the DFM-EM model with 2 and 3 factors relative to the RMSE of the AR(1): $RMSE(model, horizon)/RMSE(AR(1), horizon)$. Relative RMSEs are reported for different dates in consideration of the release date of U.S. GDP. For example, the RMSEs at 30 weeks refers to the RMSEs 30 weeks prior to the release date.

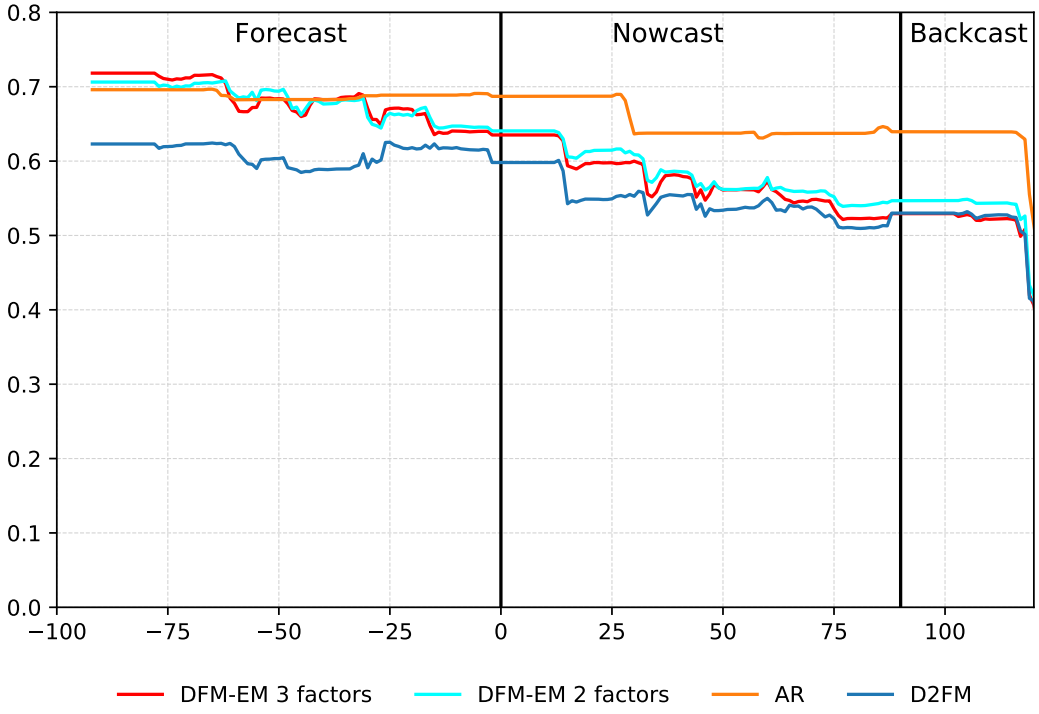


Figure 4.4: This figure reports the RMSFE evolution along the different forecasting horizons of the D^2FM model versus its competitors. The x-axis represents the difference in days between the model reference time index for that prediction and the related reference date of U.S. GDP. For example, 0 indicates the forecast made at the beginning of the current quarter, while −25 refers to a forecast made 25 days before the starting of the reference quarter.

	Forecasting	Nowcasting	Backcasting
	6 weeks	4 weeks	2 weeks
D^2FM	**0.85**	**0.83**	**0.91**

Table 4.5: Comparison of RMSEs relative to the AR(1) benchmark for monthly variables.

Notes: This table reports the average RMSFE of the D^2FM model relative to the RMSFE of the AR(1) across all monthly variables included in the model. Relative RMSEs are reported for different dates in consideration of the release date of the monthly variables. For example, the RMSEs at 6 weeks refers to the RMSEs 6 weeks prior to the release date of the variable under consideration.

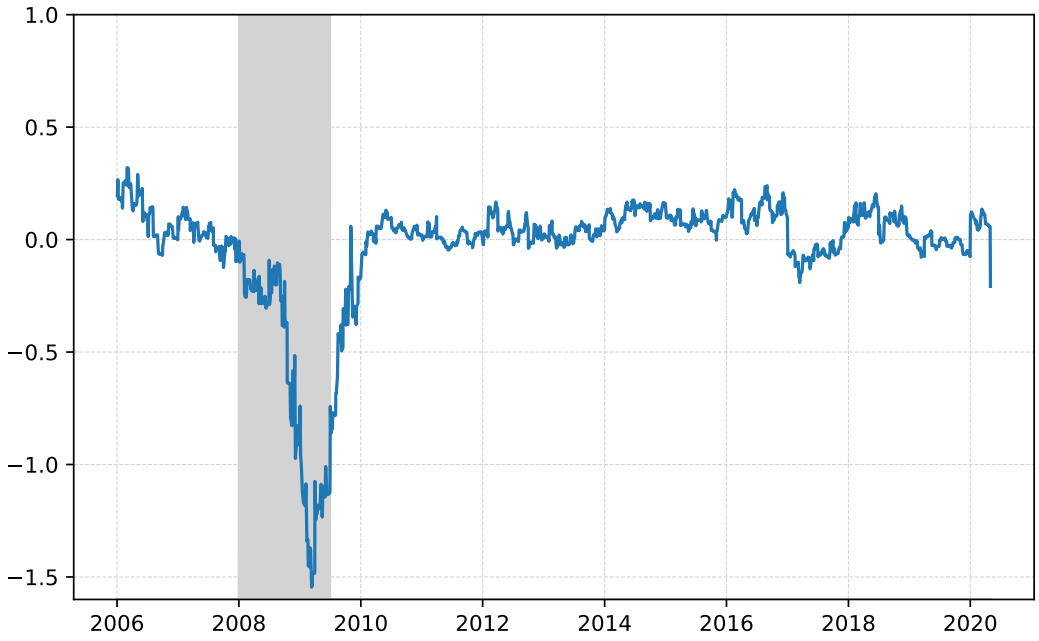


Figure 4.5: This Figure reports the Composite Indicator computed in real time using the D^2FM of Section 4.6.2. The grey area represents the financial crisis of 2008.

4.6.3 A Real-Time Synthetic Indicator of the Business Cycle

As a final exercise, we show how to build a composite indicator of the state of economy in real-time using the decoding map (or loadings). We do this by aggregating the latent states trough a weighting scheme. Specifically, we define

the composite indicator as

$$CI = \sum_{k=1}^{r} \sum_{i=1}^{N} f_k \frac{\theta_{F,k,i}^2}{||\boldsymbol{\theta}_F||_{Fr}^2} \ , \tag{4.23}$$

where f_k for $k = 1, ..., r$ are the common factors (the code) and $\theta_{F,k,i}$ is the matrix of the coefficient for the factor k at variable i, as in the Equation 4.9′, while $||\boldsymbol{\theta}_F||_{Fr}^2 = \sum_{k=1}^{r} \sum_{i=1}^{n} \theta_{F,k,i}^2$ is the squared Frobenius norm of the matrix coefficients. The sign of the indicator is fixed to have a positive correlation with GDP. Figure 4.5 reports the composite indicator using the real time out of sample exercise, that is shown to track well the developments in the US economy. Thus, the indicator could be potentially combined with others such as the one introduced in *Lang et al.* [276] to provide assessments of the current phase of the business cycle.

4.7 Conclusion and Future Research

The central contribution of our work is to introduce Deep Dynamic Factor Models (DDFM or D^{2}FMs) by showing how to embed autoencoders in a dynamic nonlinear factor model structure with idiosyncratic components, and accounting for mixed frequencies and missing data. Following the discussion on interpretable models in Section 3.4, the D^{2}FM is interpretable at least in two dimensions relevant to real time macroeconomic forecasting. In particular, the so called *news* and *impact* [39] can be computed in the proposed framework as it is currently done in DFMs. This allows for an explanation of the change in the conditional prediction of a target variable given the new information set. Moreover, similarly to DFMs the lower dimensional space of common components learned by the model can be interpreted as features of the Busyness Cycle which can be summarised in a composite indicator as shown in the empirical section of this chapter.

The empirical applications on synthetic and real data show the potential and generality of such a methodology. Possible extensions of the approach are numerous and different. For example, the model capability can be fur-

ther expanded by including nonlinear dynamic equations. Also, it could be interesting to attempt modifications of the objective function to allow for a quantile approach [277, 278]. Furthermore, one could explore the usefulness of alternative data (e.g., text data, satellite images) to the model prediction performance.

Many other extensions are possible. For example, the model could be used for asset allocation and portfolio optimisation. In Chapter 6 we will attempt this. In fact, we will compare and combine the approach with other classical financial and newer financial deep learning methods for portfolio optimisation.

Ultimately, a final note on interpretability. Indeed, while on-line updates of the latent states given the observables are directly interpretable via computations of *news* and *impact* [93] when applying filtering techniques, further computations are needed to fully explain what the model has learned during off-line training. We discuss this in detail in Appendix A.1.

Chapter 5

A Neutral Baseline for Shapley Values in MLPs

It is true that some models are intrinsically interpretable, whereas others require the user to develop auxiliary methods to explain what the model in use is capturing and why it is suggesting to take a given choice. In Section 3.4.3 of the methodological background chapter we discussed feature attribution methods and the Shapley values, and importantly, we highlighted that the definition of a baseline (aka reference point) indicates one limitation. The common practice in choosing a baseline for Shapley values is to use the vector of zeros [240, 239, 230, 236], which coincides with the average baseline when features are standardised. However, such a generic choice does not fit all applications. For instance, in a classification task, with binary features representing the presence or absence of an entity, given an example and its prediction value, such a baseline would always measure a null contribution for each feature with value equal to zero. Ill-defined baselines can drastically change the interpretation of these Shapley values. The use of improper baselines causes issues when interpreting the contribution of features. In particular, most of the baselines proposed for attribution methods are selected independently with respect to the model. Thus, they do not explain a prediction made by the model, but rather a difference with respect to some arbitrary value taken by the model. In this chapter we aim to solve this issue by leveraging the concepts of *neutrality value*

and *fair baselines*. The neutrality value is used as a parameter to represent the point at which a user of a machine learning model is unsure about whether to take one decision rather than another. For example, this could be the probability of default at which a bank is unsure about whether to approve or reject a loan request. The fairness of the baseline choice is intended to avoid biasing the importance of a feature with respect to another due to distributional differences in those variables. Using these two concepts as our starting point, we theoretically demonstrate the existence of at least a *neutral baseline* that is *fair*, given a dataset and a model. Here, we develop an algorithm to search for this baseline. Using synthetic data and a real dataset from the financial domain, we then empirically demonstrate the ability of this baseline to have higher explanability power than other choices of baseline.

5.1 Feature Attribution Methods

Feature attribution methods have been introduced in Section 3.4 when discussing explainability in Deep Learning. They are used to indicate how much each feature contributes to the prediction for a given example. In 3.4.3 we saw that a theoretically grounded feature attribution method is provided by the Shapley values and its approximations [231, 236]. When explaining a prediction with Shapley values, we need to perform two steps. First, we need to define a *baseline*, then with an example we compute the Shapley value of each of the model's features.

For a neural network, $G_{\boldsymbol{\theta}}(\cdot)$ with parameters $\boldsymbol{\theta}$, and K input features, the contribution of feature j calculated according to the Shapley value for input $\mathbf{x} = [x_1, x_2, \ldots, x_K]$ is given by:

$$\sum_{S \subseteq P \setminus \{j\}} \frac{|S|!(|P| - |S| - 1)!}{|P|!} \left(G_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{S \cup \{j\}}) - G_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_S) \right), \tag{5.1}$$

where P is the collection of all feature indices, element i of vector $\tilde{\mathbf{x}}_S$ is given by $\tilde{x}_{S,i} = x_i \mathbb{1}_{\{i \in S\}} + b_i \mathbb{1}_{\{i \notin S\}}$ (similarly for $\tilde{x}_{S \cup \{j\}}$), and b_i is the baseline value for feature i. The *baseline* models the missingness of a feature, i.e., it replaces

that feature when it is absent. As argued by *Sturmfels et al.* [237], the concept of missingness is not well defined or explored in ML models. Alternatively, feature(s) can be removed from the model via marginalisation, thus assuming a distribution for b_i. Nevertheless, the standard practice in setting up a baseline is to assign a vector of zeros for all features. However, this choice might provide wrong interpretations and even give zero importance to significant features.

The simplest choice of a baseline is the *zero* vector baseline [240, 230, 239, 236], which coincides with the *average* vector baseline when features are standardised. However, this choice could be misleading. For instance, consider a feature in a model that is most significant when its value is zero. Now, if we compute the Shapley values on this model with the zero baseline, the importance of that will be zero. One way of addressing the zero-baseline insensitivity problem is to use the maximum distance baseline (*mdb*) method [237]. This baseline consists of taking the furthest observation from the current one in an L^1 norm. However, this approach unequivocally creates incoherent explanation of model outputs due to the dependence of the baseline to the underlying dataset and the specific sample explained.

Alternatively, one can sample a baseline from a multivariate distribution such as Uniform [237] or Gaussian [241]. This approach can be improved by considering a sample of baselines and by averaging the attributions computed over each baseline [241, 231, 242, 243, 244]. Another form of sampling is the one performed using the dataset. Hence, one can use the underlying empirical distributions of the dataset [231, 244, 237]. We denote this as the $p_{\mathcal{X}}$ baseline method. However, the $p_{\mathcal{X}}$ baseline increases the computational cost of estimating feature attributions linearly with respect to the number of draws. Moreover, this choice of baseline does not allow the setting of a reference value on the model output when computing the Shapley values. This is important when decisions are taken with respect to a specific value of the model. This generalises also to the other baselines described.

5.2 The Neutral Baseline

We begin by theoretically justifying the existence of a baseline according to well defined concepts of neutrality value and fair baselines, and then presenting the algorithm to find these baselines in multilayer perceptrons (MLPs). The intuition behind such baselines is better explained by two examples.

Example 1 (Neutrality)**.** *Consider a linear classifier used by a banking agent to predict the probability of default of clients based on two features, formally:* $G_{\boldsymbol{\theta}}(\mathbf{x})$*, with* $\boldsymbol{\theta} = [\phi_0, \phi_1, \phi_2]$ *and* $\mathbf{x} \sim \mathcal{N}(0, I_{2\times 2})$ *and* $G(\cdot)$ *being the sigmoid function; while,* 0.5 *being the threshold level for the agent to take a decision of whether to offer a loan or not, namely reject if* $G_{\boldsymbol{\theta}}(\mathbf{x}) > 0.5$*. Assume now that* $\phi_0 > 0$*, so that the decision boundary of the classifier does not cross the origin. Imagine now that a client, whose feature values are equal to an average clients' feature values, complains about a rejected loan application, and the banking agent wants to explain the decision of the classifier using Shapley values with a vector of zeros as a baseline. In this case, the agent is not going to be able to provide a meaningful explanation since the Shapley values will all be equal to 0. However, if the baseline is chosen on the decision boundary of the classifier, the Shapley values, thus calculated, will provide an explanation consistent with the loans rejection choice. Although, this example describes a client with very specific feature values, misleading attributions of importance occurs every time the client's feature values lie between the baseline of zeros and the decision boundary of the classifier. Indeed, if for example, we set* $\phi_0 = \phi_1 = \phi_2 = 1$ *then* $G_{\boldsymbol{\theta}}(x_1 = -0.25, x_2 = -0.25) = 0.62$*, but the Shapley values with zero baseline for* x_1 *and* x_2 *are negative, and thus provide an inconsistent justification for the decision taken by the classifier (reject the loan).*

In this work, we follow *Bach et al.* [137] and argue that the baseline should rely on the decision boundary of the classifier. If we return to the above example, and the baseline had been on the decision boundary, then the Shapley values would have been different from zero and with positive values for those features causing the loan to be rejected. We relate the concept of missingness

in Shapley values to the output of the model through Definition 4.

Definition 4 (Neutrality Value)**.** *Given a model prediction $\hat{y}$ and a decision maker, we say that the value α is neutral if the decision maker's choice is determined by the value of $\hat{y}$ being either below or above α.*

This neutrality value is usually set by the decision maker. For example, consider the probabilistic classifier used in the loan issuing example above. In that context the banking agent is happy to approve a loan if the client has less than a 50% chance of default; hence the neutrality value is equal to $\alpha = 0.5$. Hypothetically, if the model's output is 0.5, then the agent will be indecisive about issuing the loan, as the model has no relevant information to offer to the agent in order to make a decision. We argue that when this is the case for the model, then the same standard should be used for the explainability approach applied to the model.

The central idea is that this neutrality value can lead to a point in the input domain that could be used as a baseline. However, given a neutrality value and a single-layer perceptron (SLP) with more than one (continuous) input feature, there are an infinite number of possible combinations of such inputs that lead to the same neutral output. We narrow down the solutions by introducing the concept of *fairness*: we say that $\mathbf{x}$ is in the space of *fair* baselines for a monotonic model $G_{\boldsymbol{\theta}}(\cdot)$ if every element x_i with a positive monotonic relation in $G_{\boldsymbol{\theta}}(\cdot)$, when sorted with respect to all of its possible realisations, has the same number of such realisations to its left with respect to those on the right of any other element of $\mathbf{x}$ with negative monotonic relation in $G_{\boldsymbol{\theta}}(\cdot)$, and to the one on the left of any element of $\mathbf{x}$ with positive monotonic relation in $G_{\boldsymbol{\theta}}(\cdot)$.[1] We call such baselines fair, as we are being fair in representing each feature in relation to its probability space, and given its relation with the model. The following simple example shows why this is a desirable property.

[1]To note that the monotonicity of the model is intended with respect to each single input, that is the monotonicity of $G_{\boldsymbol{\theta}}(\cdot)$ is meant to be element by element.

Example 2 (Fairness)**.** *Consider the same model of Example 1 with the simplifying assumption* $\boldsymbol{\theta} = [1, 1, 1]$*. Both the baselines* $\boldsymbol{b}^1 = [-0.5, -0.5]$ *and* $\boldsymbol{b}^2 = [-1.5, +0.5]$ *are neutral, but only the first satisfies our definition of fairness, i.e.,* $b_1^1 = C_1^{-1}(0.69)$ *and* $b_2^1 = C_2^{-1}(0.69)$*, where* $C_i(\cdot)^{-1}$ *is the inverse cumulative distribution function of* x_i*. Let's define* $Sh_{\{G_\theta;\boldsymbol{b}\}}(x_i)$ *as the Shapley value of feature* i *with respect to the baseline* $\boldsymbol{b}$ *and the model* $G_\theta(\cdot)$*. Because of the assumptions made on the data generating process and the model, the two features must have the same expected importance. We derive:*

$$\begin{aligned} &E_{x_1,x_2}[(Sh_{\{G_\theta;\boldsymbol{b}\}}(x_1) - Sh_{\{G_\theta;\boldsymbol{b}\}}(x_2))] = \\ &E_{x_1}[G_{\boldsymbol{\theta}}(x_1, b_2)] - E_{x_2}[G_{\boldsymbol{\theta}}(b_1, x_2)] = 0 \end{aligned} \tag{5.2}$$

and it is clear that only the fair baseline $\boldsymbol{b}^1$ *satisfies the equality.*

The following definition formalises fairness.

Definition 5 (Space of Fair Baselines)**.** *Consider a dataset in* $\mathbb{R}^k$*,* $k \geq 1$*, generated by a distribution. We then define the set of fair baselines for a monotonic model* $G_{\boldsymbol{\theta}}(\cdot)$ *as*

$$\tilde{B} = \{\mathbf{x}^p \in \mathbb{R}^k : x_j^p = C_j^{-1}(\mathbb{1}_{\theta_j > 0} \cdot p + \mathbb{1}_{\theta_j < 0} \cdot (1-p)), \quad p \in [0,1],\, j = 1, 2, \ldots, k\}, \tag{5.3}$$

where C_j^{-1}*s are inverse marginal CDFs.*

Alternatively, we say that $\mathbf{x} \in \mathbb{R}^k$ is a fair baseline, if:

$$\mathbf{x} \in \tilde{B} \iff C_j(x_j) = \begin{cases} C_i(x_i) & \text{if} \quad sign(\theta_j) = sign(\theta_i) \\ 1 - C_i(x_i) & \text{if} \quad sign(\theta_j) \neq sign(\theta_i) \end{cases} \forall j, i = 1, \ldots, k. \tag{5.4}$$

Using the two definitions, neutrality value and space of fair baselines we next demonstrate the existence of a fair baseline that when given to an MLP returns the neutrality value. Before we begin, we need to state the following two assumptions:

A1. All activation functions are monotonic and continuous.

A2. All marginals cumulative distribution functions (CDFs) of the joint CDF of the input features are bijective and continuous.

Assumption **A1** is reasonable since many activation functions are monotonic like linear, sigmoid, tanh, softplus, ReLU, LeakyReLU and ELU. All of these functions are continuous. **A2** is instead a technical assumption required in the proof of our results later.

5.2.1 The SLP Case

Using Definitions 4 and 5, **A1** and **A2**, the following proposition guarantees the existence of a neutral fair baseline for SLPs:

Proposition 1 (Existence of a Neutral Fair Baseline for SLPs)**.** *Given an SLP* $G_{\boldsymbol{\theta}}(\cdot)$ *satisfying* ***A1****, a dataset satisfying* ***A2****, and a neutrality value* α *in the image of* $G_{\boldsymbol{\theta}}(\cdot)$*, then there exists at least a fair baseline* $\mathbf{x}$ *such that* $G_{\boldsymbol{\theta}}(\mathbf{x}) = \alpha$*.*

If we replace **A1** with the following more stringent assumption, the solution becomes unique (see Fig. 5.1 for intuition).

A1'. All activation functions are strictly monotonic and continuous.

Proposition 2 (Uniqueness of a Neutral Fair Baseline for SLPs)**.** *Given an SLP* $G_{\boldsymbol{\theta}}(\cdot)$ *satisfying* ***A1'****, a dataset satisfying* ***A2****, and a neutrality value* α *in the image of* $G_{\boldsymbol{\theta}}(\cdot)$*, then there exists a unique fair baseline* $\mathbf{x}$ *such that* $G_{\boldsymbol{\theta}}(\mathbf{x}) = \alpha$*.*

The proof of Proposition 1 provided in Appendix B suggests a way to find one fair baseline for an SLP. This is formalised in Algorithm 3. This algorithm using empirical CDFs instead of theoretical ones requires as inputs an SLP $G_{\boldsymbol{\theta}}(\cdot)$, a neutrality value ($\alpha$), a quantile function ($Q_j$, obtained from the marginal empirical CDF) for each dimension of input features, a granularity level $\delta > 0$, and a tolerance level $\epsilon > 0$. The δ and the ϵ control the speed of search and the margin of error in finding a baseline such that $|G_{\boldsymbol{\theta}}(\mathbf{x}) - \alpha| < \epsilon$. This algorithm starts the search from the lowest possible output value of the SLP, which is when $p = 0$, and stops when it reaches a point which is close

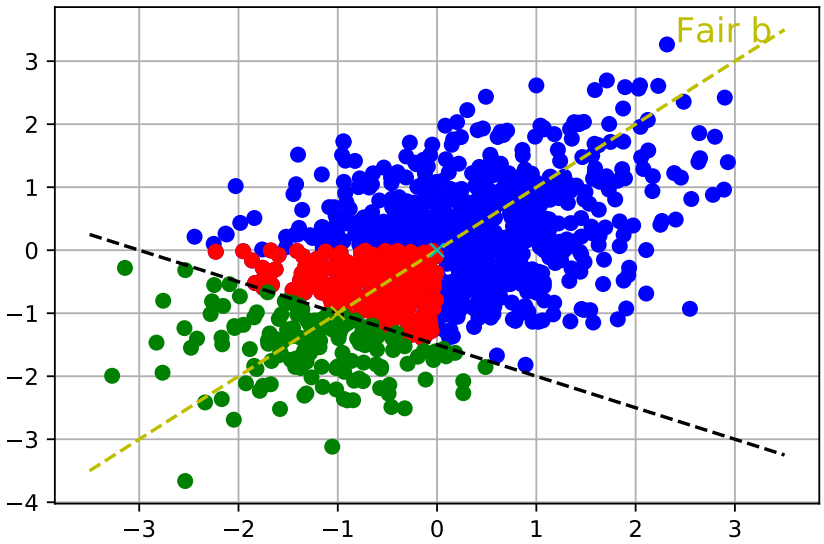


Figure 5.1: The chart shows a linear binary classifier with two mean zero input features and positive intercept term. The black dashed line represents the decision boundary (all the points on this line are such that the output of the model is exactly 0.5). Green circles are negative labels. Blue circles represent positive labels. Red circles represent positive labels with misleading Shapley values for both features when using the zero (or average) baseline. The yellow dashed line indicates the set of fair baselines.

enough to α. This is possible because by using the parameters of the SLP, we can restrict and define an order in function of p for the set of fair baselines (as in Line 2). This allows us to test these baselines from the smallest SLP value to the largest.

Algorithm 3 Neutral Baseline Search for SLPs

Input: $G_{\boldsymbol{\theta}}(\cdot), \alpha, Q, \delta, \epsilon$
Ouput: $\mathbf{b}$

1: $p \leftarrow 0$
2: $\mathbf{b} \leftarrow [Q_j(\mathbb{1}_{\theta_j>0} \cdot p + \mathbb{1}_{\theta_j<0} \cdot (1-p))\ \forall j \in 1,..,K]$
3: **while** $G_{\boldsymbol{\theta}}(\mathbf{b}) - \alpha < \epsilon$ **do**
4: $\quad p \leftarrow p + \delta$
5: $\quad \mathbf{b} \leftarrow [Q_j(\mathbb{1}_{\theta_j>0} \cdot p + \mathbb{1}_{\theta_j<0} \cdot (1-p))\ \forall j = 1,..,K]$
6: **end while**

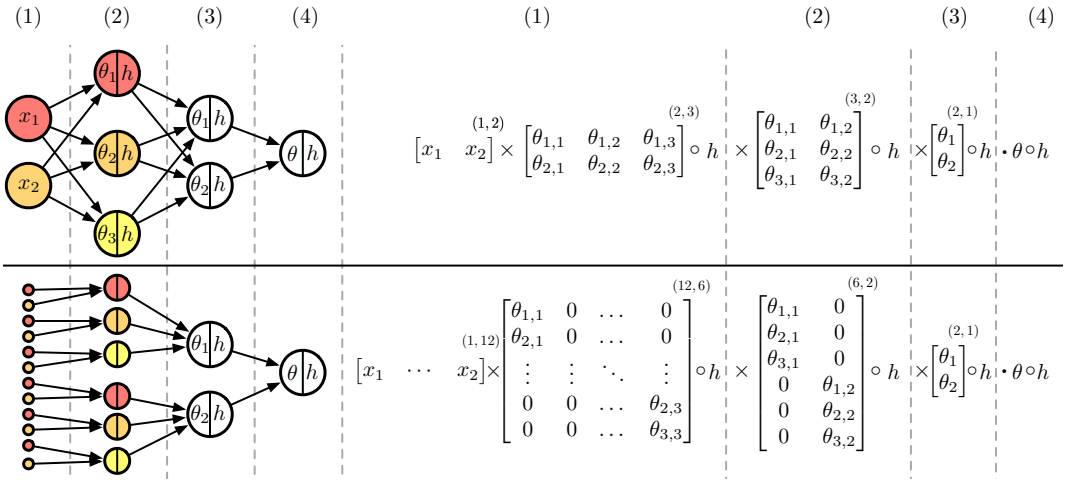


Figure 5.2: An MLP (above) and its equivalent sparse representation (below).

Algorithm 4 Neutral Baseline Search for MLPs

Input: $G_{\boldsymbol{\theta}}(\cdot), \alpha, Q, \delta, \epsilon$
Ouput: **b**

$queue \leftarrow []$
enqueue$([\alpha], queue)$
$T = 1$
for $l = L : -1 : 2$ **do**
  for $t = 1 : T$ **do**
    $\alpha \leftarrow$ dequeue$(queue)$
    for $j = 1 : |\alpha|$ **do**
      $\text{SLP}_{(l,j)} = G^{(l)}(G_{\boldsymbol{\theta}}^{(l-1)} \cdot \theta_j^{(l)})$ ▷ SLP Building
      $\tilde{\alpha} \leftarrow$ **Algorithm 1**$(\text{SLP}_{(l,j)}, Q^{l-1}, \alpha_j, \delta, \epsilon)$
      enqueue$(\tilde{\alpha}, queue)$
    end for
  end for
  $T = T * k_{l+1}$
end for
while queue is not empty **do**:
  $\alpha \leftarrow$ dequeue$(queue)$
  $\mathbf{b} \leftarrow \mathbf{b}||\alpha$
end while

5.2.2 The MLP Case

Finding a baseline for MLPs is more complicated, because there is no easy way to order the baselines in function of p, as in the SLP case, unless the MLP is monotonic in each of the features.

We observe that an MLP with L layers can be decomposed into $\sum_{l=2}^{L}\prod_{l'=l}^{L} k_{l'}$ SLPs by replicating every node at layer l, k_{l+1} times, i.e., the number of nodes at layer $l+1$, and considering every node in the layer $l+1$ as an SLP with input given by the layer l. Based on this observation, we can recursively apply Algorithm 3 backwards through the layers of the model to recover the neutrality values across those SLPs, from the output layer to the input layer. This will provide $\prod_{l=2}^{L} k_l$ baselines, one for each SLP in the first hidden layer. This is implemented in Algorithm 4. Note the concatenation of the found fair baselines from Line 15 to 18.

Finally, in order to aggregate these baselines, we define an equivalent sparse representation of an MLP (SparseMLP), which is constructed by concatenating each of the SLPs defined above. An example of this transformation is provided in Figure 5.2. We find this necessary because to compute the Shapley value for each example-feature pair we can now use all fair baselines found with Algorithm 4 at once.[2]

5.2.3 Speeding up the Search

If features are approximately normally distributed, we can use conventional convex optimisation packages to be more efficient. Thus, assuming the inputs are approximately Gaussian (with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, the vector of means and standard deviations), we can replace the search method of Algorithm 3 with the following (constrained) optimisation problem:

$$\underset{\tilde{p}}{\operatorname{argmin}} \left[G_{\boldsymbol{\theta}}(\tilde{p}\boldsymbol{\sigma} + \boldsymbol{\mu}) - \alpha\right]^2 \tag{5.5}$$

which can be augmented with constraints on the input space. The argument $\tilde{p}$ in the minimisation problem is a scalar that mimics the p of Algorithm 3. However, this trick makes it possible to calculate the derivatives of $G_{\boldsymbol{\theta}}(\cdot)$ with

[2]To note that this equivalent sparse representation is not unique. As one can permute the order of the SLPs and generate another equivalent sparse representation. This implies that even when the baselines found are unique, the overall approach is unique only up to a permutation. However, being all of these representations observationally equivalent, this does not have any practical impact.

respect to $\tilde{p}$, allowing us to use conventional convex optimisation methods. To then obtain the baseline vector in the input space we need to set $\mathbf{b} = \tilde{p}\boldsymbol{\sigma} + \boldsymbol{\mu}$. Note that if there are negative elements in $\boldsymbol{\theta}$, we need to multiply those and the related features by -1 before running the optimisation; the final solution is then retrieved by repeating this sign swap on the respective element in the baseline vector found.

5.3 Experiments

We now evaluate in a classification task the explanability power of *zero*, *average*, $p_{\mathcal{X}}$, and *mdb* baselines, against our proposed baseline method ($neutral_{\alpha}$), where we set the neutrality value α to 0.5 being it a binary classification context.[3]

To this purpose we use two datasets: A less realistic dataset (synthetic), where we aim to compare the choice of baselines by limiting the drawbacks of the employed measures, and a real dataset, where we seek to validate these results in a more realistic scenario. In what follows, we first present the setup of evaluation methods used to quantify explainability power, and thereafter we present the datasets and training setup. Lastly, we present and discuss the results.[4]

5.3.1 Evaluation Measures

To evaluate the local explanability power we use both ROAR [248] and a perturbation test similar to the one introduced by *Ancona et al.* [249]. Indeed, since we are evaluating probabilistic classifiers, rather than measuring deviations between predictions, we prefer a more appropriate measure of confidence: the *absolute logits*. Absolute logits are defined as $|\log(G_{\boldsymbol{\theta}}(\mathbf{x})/(1-G_{\boldsymbol{\theta}}(\mathbf{x})))|$, where $G_{\boldsymbol{\theta}}(\cdot)$ is a probabilistic classifier and $\mathbf{x}$ is an example. These absolute logits can be used to measure prediction confidence; in fact, when these increase it means that the confidence of the model in predicting one of the two classes is

[3] One could actually estimate the optimal threshold, see for example *Alessi and Detken* [279] with respect to financial early warning indicators.

[4] The code used to run these experiments is available at the following weblink: `https://github.com/cosimoizzo/Neutral-Baseline-For-Shapley-Values`.

increasing and the opposite is true otherwise.

Our preference for this measure is also supported by information theory. Standard logits can be seen as the difference between two Shannon information. Since in this circumstance Shannon information represents the confidence level of the model in classifying the instance as positive or as negative, we take the absolute value of the standard logits to measure the variation in Shannon information in both directions. As this measure decreases (increases), the Shannon information decreases (increases). Furthermore, when the Shannon information content is 0, we can argue that the model becomes uninformative.

The combination of ROAR with our perturbation test allows us to test feature attribution approaches in two dimensions. The first is with respect to the ranking of the features based on their importance, and the correctness of this is tested with ROAR. However, ranking features in order of importance is not the only purpose of an explainability method. It is also useful to check whether the actual values provided by the explainability method are consistent with the human interpretation of the model output. This second dimension is at least partially assessed with the absolute logits measure. Indeed, by using this measure, we are assessing whether when removing features in order of importance we are actually reducing the information content that the model can transfer to the human user in support of a given choice.

5.3.2 Datasets and Training Details

Next, we present the two datasets. We use a synthetic dataset to simulate a scenario with independent features and controlled feature importance (the independence allows us to avoid drawbacks associated with some evaluation methods). The real dataset is used to experiment with a real use case.

Synthetic dataset. We generate a dataset with 5 independent features governed by the following sampling process. Given N, the number of examples $\mathbf{x}$ we want to generate, we sample each $\mathbf{x}$ from a multivariate normal distribution with $\mathbf{0}$ mean and covariance matrix $I_{5\times5}$ to guarantee that features are

independent among each other:

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, I_{5\times 5}). \tag{5.6}$$

We then define the importance of each feature by sampling a vector $\boldsymbol{\phi}$:

$$\phi_0 = \mu_0 \quad \text{with} \quad \mu_0 \sim \mathcal{U}(-15, 15) \tag{5.7}$$

$$\phi_j = (-1)^{\sigma_j} \cdot \mu_j \quad \text{with} \quad \sigma_j \sim \mathcal{B}ern(0.5),\ \mu_j \sim \mathcal{L}og\mathcal{N}(j, 1),\ j = 1, \ldots, 5 \tag{5.8}$$

where ϕ_0 is sampled from a uniform distribution and ϕ_j is decomposed into two components, a sign and a magnitude. The sign is sampled from a Bernoulli distribution with $p = 0.5$, while the magnitude is sampled from a log-normal distribution with mean j and variance 1. This makes, in expectation, the feature with the larger j more important than the one with a lower j.

We then add a constant term to each $\mathbf{x}$ to include a bias term and multiply this new vector to $\boldsymbol{\phi}$:

$$y^* = \boldsymbol{\phi}'([1]||\mathbf{x}) \tag{5.9}$$

At this point we have generated $\mathbf{x}$ and y^* values. Since we are generating a synthetic dataset for a binary classification task, we define the label values (y), by transforming y^* as follows:

$$y = \begin{cases} 1 & \text{if } Q(\tau_1) < y^* < Q(\tau_1 + \tau_2) \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

where $\tau_1, \tau_2 \sim \mathcal{U}[0.3, 0.5]$ and Q is the quantile function of the empirical CDF of y^*, which allows us to control the balance between positive and negative classes (i.e., by ensuring that at least 30% of the observations are in each of the two classes), and makes this dataset unsolvable by a linear classifier.

Credit Card Default dataset. This is a dataset about default of credit card clients [280]. The dataset contains one binary target variable and 23 features. The number of observations is 29351. To apply ROAR to such dataset, we would

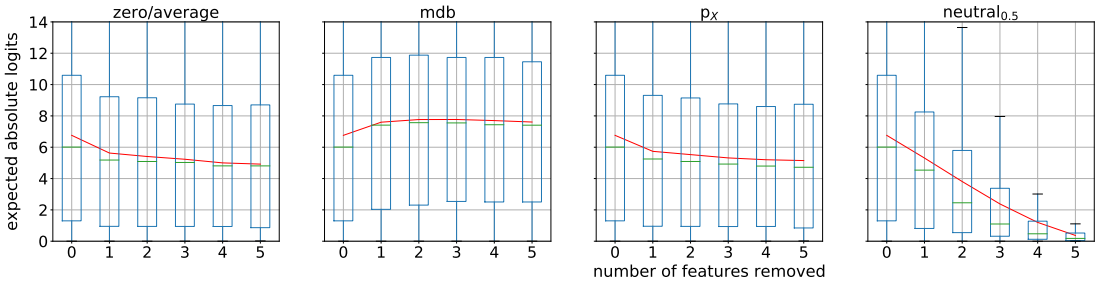
need to compute Shapley values on all of the samples, which is computationally demanding.[5]. Therefore, we reduce the number of observations to 300 for computational reasons. We do so by sampling these observations while keeping the two classes balanced, that is we randomly draw 300 samples from the original 29351 while imposing the constraint that half of the sampled target labels should be 0 and the other half should 1. This is the portion of the dataset on which we train the model and carry out the analysis. Morevoer, to reduce even further the computational cost of Shapley, this time with respect to the number of features dimension, we compute them using sampling [233].

Before training the model, both datasets are preprocessed: the synthetic one by a simple standardisation, and the real one by using a min-max scaler. The activation function is always a sigmoid. All MLPs are trained with binary cross-entropy loss function using ADAM as optimiser with its default parameters, except that the learning rate is increased to 0.05. We use an early stopping criteria to avoid overfitting, where we stop training after the loss on the validation set has not improved over 3 consecutive epochs. The real dataset is divided into 60% training set, 20% validation set, and 20% test set. The synthetic dataset consists of 600 observations for training (80%) and validation (20%), plus another 100 observations for the test set. The rest of the hyper-parameters, i.e., the number of hidden layers and number of neurons in each layer, are chosen via a random sampling of models using the training and validation sets. We sample 300 models with number of hidden layers from 0 to 5, and number of neurons for each hidden layer from 1 to 10. The results obtained on the synthetic dataset are based on 100 Monte Carlo (MC) simulations, where in each simulation a new synthetic dataset is generated and, on it, training and validating the best MLP as mentioned above. Thereafter, we evaluate the attribution method with the various choice of baselines on a test set of 100 observations.

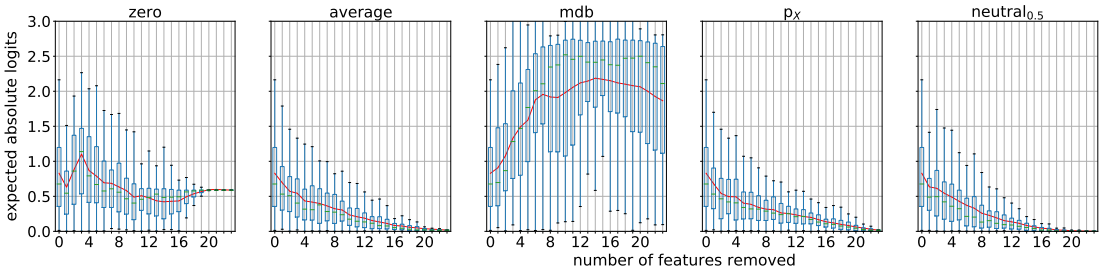
[5]It would mean calling the Shapley values method on each of the almost 30 thousand samples, this for each iteration of ROAR.

5.3.3 Results and Discussion

Figures 5.3a, 5.3b show evolutions of expected absolute logits when removing features in order of importance. We show both average values and box-plots, where for synthetic datasets both of them are computed across test set observations and MC simulations, while for the real dataset only across test set observations. The only baseline that in all datasets guarantees a monotonic decrease to zero in the information content of the model when removing features is the $neutral_a$.



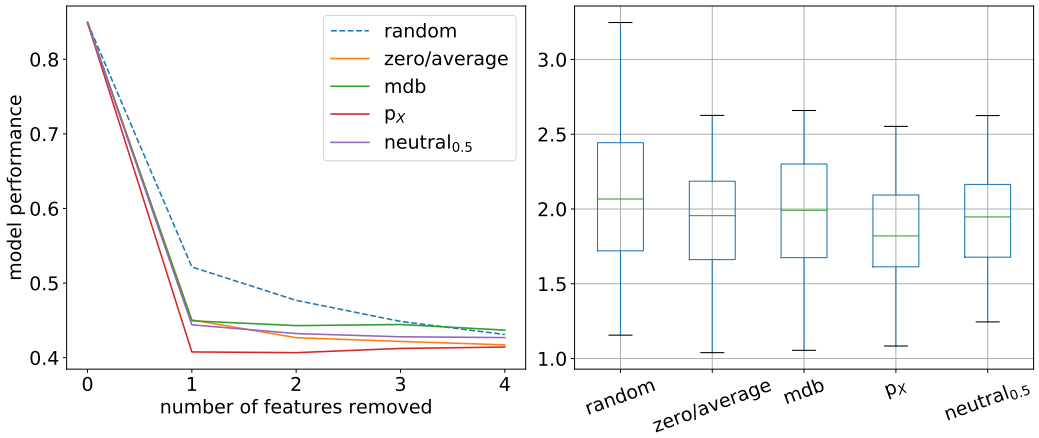
(a) Synthetic dataset.



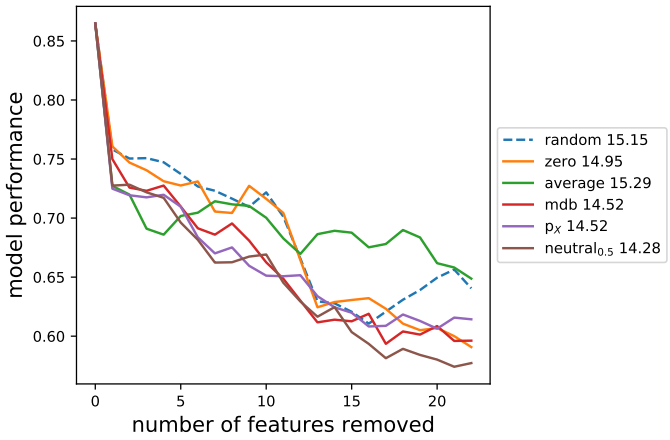
(b) Credit card dataset.

Figure 5.3: Information content on the synthetic and the credit card dataset.

Figures 5.4a, 5.4b show ROAR curves and scores, where the latter are computed as the area under the related curves. Intuitively, the lower is this area the better is the method in identifying the most important features first on a per example basis, as the performance of the model is deteriorating (decreasing) faster. For the synthetic datasets we also compute box-plots of these scores across the 100 MC simulations together with the average curves. Although in the synthetic dataset $p_{\mathcal{X}}$ achieves the best score followed by *zero* and $neutral_a$, in the real dataset the $neutral_a$ baseline achieves the best score. Since $p_{\mathcal{X}}$ and $neutral_a$ show similar ROAR scores, the two approaches do equally well in



(a) Synthetic dataset.



(b) Credit card dataset.

Figure 5.4: ROAR on the synthetic and the credit card dataset. Panel (a) shows the average curve and a box-plot of the area under the curves across the 100 MC simulations. Panel (b) reports the area under the curve in the legend.

ranking features in order of importance.

5.4 Conclusion

In this chapter we have investigated the identification of baselines for Shapley values based feature attribution methods and MLPs. We have introduced the concept of neutrality and fair baselines. Their combination has allowed us to develop a neutral baseline that better explains model based decisions compared to other baselines analysed.

It is important to mention that the computational cost of searching the

$neutral_a$ baseline increases exponentially with respect to the number of hidden layers. Also, in this chapter we did not analyse how to apply such method to recurrent networks and how to extend it to regression problems. With regard to regression problems, we provide a financial example in Chapter 6.[6] Meanwhile, we leave it to future analysis to devise on possible solutions to the computational issue and to the extensions for other types of network architectures.

[6]In Appendix A.1 we discuss how to apply the methodology to the D^{2}FM estimated latent sates.

Chapter 6

Portfolio Optimisation and Deep Learning

Following the discussion from Section 3.3 about portfolio optimisation and utility function, in this chapter we introduce and empirically evaluate a way to apply deep learning techniques via direct deterministic policy gradient methods to different reward formulations that depend on the desired degree of approximation of an investor's utility function. We compare against other deep learning and standard financial approaches, such as mean-variance, minimum-variance, risk-parity, and hierarchical risk parity which have been discussed in details in Section 3.3 of the background chapter. Moreover, since this problem can be decomposed into a prediction and an optimisation problem conditioned on such predictions, we also combine all the approaches analysed with the D^2FM (or DDFM) introduced in Chapter 4 to deal with the forecasting and the state estimation problem. A comparison of these methods is carried out using daily data from the U.S. stock market, while in the related appendix (i.e., Appendix C) we perform a robustness check on weekly data. We conclude this chapter by carrying out an analysis of the relation between the excess returns generated by selected strategies, and standard financial factors. We do so both by using linear regression methods and deep learning techniques together with Shapley values and the neutral baseline developed in Chapter 5.

6.1 Dynamic Portfolio Optimisation with Deep Learning

In this section we introduce our formulation of dynamic portfolio optimisation, which is adapted from the portfolio rebalancing framework presented in *Sun et al.* [217]. We assume there is no consumption and income, and thus the wealth process in this circumstance is called self-financing [110]. Let's start by casting the dynamic portfolio optimisation problem into a Markov Decision Process (MDP). Namely, we assume that at time t the investor observes the realisation of a stochastic vector $\mathbf{x}_t$ which includes the period t asset returns $\boldsymbol{r}_t$, the previous period portfolio allocation weights $\boldsymbol{w}_{t-1}$, and any other relevant information to correctly specify the state transition kernel from the current state to the next state. Conditioned on the realisation of the current state $\mathbf{x}_t$, the investor chooses the new allocation weights $\boldsymbol{w}_t = \pi_\theta(\mathbf{x}_t)$ via the parametric policy function π_θ in order to maximise the value function. To this purpose, we can express the value function of the investor as follows

$$J_t(\mathbf{x}_t, \pi_{\boldsymbol{\theta}}) = \mathrm{E}_t[G(\mathbf{x}_t, \pi_{\boldsymbol{\theta}}) + J_{t+1}(\mathbf{x}_{t+1}, \pi_{\boldsymbol{\theta}})] \tag{6.1}$$

where E_t is the expectation operator conditioned on information up to time t which is summarised into $\mathbf{x}_t$. The first term on the right hand side of equation (6.1) is the instantaneous reward and it is defined here as

$$\mathrm{E}_t[G(\mathbf{x}_t, \pi_{\boldsymbol{\theta}})] = C(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t), \boldsymbol{w}^+_{t-1}) + \mathrm{E}_t[U(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t)'(1 + \boldsymbol{r}_{t+1}))], \tag{6.2}$$

where $C(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t), \boldsymbol{w}^+_{t-1})$ represents transaction costs incurred when taking action $\pi_{\boldsymbol{\theta}}(\mathbf{x}_t)$ from state $\mathbf{x}_t$. The latter includes the previous allocation weights $\boldsymbol{w}_{t-1}$ and the current realised returns $\boldsymbol{r}_t$ through which it is possible to compute the end of period allocation weights $\boldsymbol{w}^+_{t-1} = \frac{\boldsymbol{w}_{t-1} \circ (1+\boldsymbol{r}_t)}{\boldsymbol{w}'_{t-1}(1+\boldsymbol{r}_t)}$.[1] Thus, equation (6.2)

[1] In the empirical section we specify these transaction costs to be quadratic in order to allow differentiability everywhere, as we will be using a policy gradient method. Thus, we set them as follows $C(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t), \boldsymbol{w}^+_{t-1}) = 0.0005 * (\pi_{\boldsymbol{\theta}}(\mathbf{x}_t) - \boldsymbol{w}^+_{t-1})'(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t) - \boldsymbol{w}^+_{t-1})$. This term

imposes a separability condition between transaction costs and instantaneous utility similarly to *Sun et al.* [217].

The second term of equation (6.2), that is $\mathrm{E}_t[U(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t)'(1+\boldsymbol{r}_{t+1}))] = \mathrm{E}_t[U(W_{t+1})]$ is the expected single period utility from wealth. If we then take a fourth order Taylor expansion of $\mathrm{E}_t[U(W_{t+1})]$ around some root point $\bar{W}=0$, and to make this approximation implementable we assume a CARA utility function of the form $U(W_{t+1}) = -e^{-\lambda W_{t+1}}$, we obtain

$$\begin{aligned}\mathrm{E}_t[U(W_{t+1})] &\propto \lambda M_{1|t}(W_{t+1}) - \frac{\lambda^2}{2}M_{2|t}(W_{t+1}) + \frac{\lambda^3}{6}M_{3|t}(W_{t+1}) - \frac{\lambda^4}{24}M_{4|t}(W_{t+1}) \\ &= \mathrm{E}_t[\widetilde{U(W_{t+1})}] = \mathrm{E}_t[U(\pi_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}_t})'(1+\boldsymbol{r}_{t+1}))]\end{aligned} \tag{6.3}$$

where λ is the risk aversion parameter that we set to 1 in the empirical section, while $M_{i|t}$ is the $i-th$ conditional moment to time t information, here again summarised in $\mathbf{x}_t$. This formulation allows us to have an intuitive statistical interpretation of investors' preferences with respect to relevant statistics of the portfolio return's distribution, and it encompasses the mean-variance preferences as a special case.[2] We can now replace $\mathrm{E}_t[U(W_{t+1})]$ with its approximation $\mathrm{E}_t[\widetilde{U(W_{t+1})}]$ and express the new value function from equation (6.1) as follows

$$J_t(\mathbf{x}_t, \pi_{\boldsymbol{\theta}}) = \mathrm{E}_t[\widetilde{G(\mathbf{x}_t, \pi_{\boldsymbol{\theta}})}] + \mathrm{E}_t[J_{t+1}(\mathbf{x}_{t+1}, \pi_{\boldsymbol{\theta}})], \tag{6.4}$$

where

$$\mathrm{E}_t[\widetilde{G(\mathbf{x}_t, \pi_{\boldsymbol{\theta}})}] = C(\pi_{\boldsymbol{\theta}}(\mathbf{x}_t), \boldsymbol{w}^+_{t-1}) + \mathrm{E}_t[U(\pi_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}_t})'(1+\boldsymbol{r}_{t+1}))], \tag{6.5}$$

noting that both terms depend just on the policy $\pi_{\boldsymbol{\theta}}$ and the properly constructed state $\mathbf{x}_t$. We can then express the $Q-$function of this problem when following the policy $\pi_{\boldsymbol{\theta}}$ for all subsequent actions and conditioned on the state

aims to impose a penalisation factor on re-balancing, and we intend to intepret it as such. Transaction costs for ETFs can vary significantly depending on the method used to compute them [281]. Indeed, the authors report that the iShares bid-ask spread is between 1.4 and 2.4 basis points, while their estimates are up to a possible 5% round-trip cost.

[2] See Section 3.3 of the background chapter for further details.

$\mathbf{x}$ and action $\boldsymbol{w}$ at time $t = 0$ as

$$Q_{\pi_{\boldsymbol{\theta}}}(\mathbf{x}, \boldsymbol{w}) = \mathrm{E}[\sum_{t \geq 0} \widetilde{G(\mathbf{x}_t, \pi_{\boldsymbol{\theta}})} | \mathbf{x}_0 = \mathbf{x}, \boldsymbol{w}_0 = \boldsymbol{w}]. \tag{6.6}$$

It is now possible to apply Reinforcement Learning and Deep Reinforcement Learning techniques, such as the ones discussed in Section 3.2.5, to solve this formulation of the dynamic portfolio optimisation problem. In particular, and assuming both the policy and the Q function are differentiable, one can apply direct deterministic policy gradient methods to optimise the parametric policy function $\pi_{\boldsymbol{\theta}}$ with respect to its parameters $\boldsymbol{\theta}$ via gradient ascent computed as follows

$$\Delta_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}} = \mathrm{E}[\Delta_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{x}) \Delta_{\boldsymbol{w}} Q_{\pi_{\boldsymbol{\theta}}}(\mathbf{x}, \boldsymbol{w})|_{\boldsymbol{w} = \pi_{\boldsymbol{\theta}}(\mathbf{x})}], \tag{6.7}$$

where the actual value $\Delta_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$ is replaced by its sample estimates $\widehat{\Delta_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}}$ which involve the computation of the derivatives of the sample estimates for the moments introduced in equation (6.3) and entering here via equation (6.6).

6.2 An Application to the US Stock Market

In this section we compare different approaches on a dataset of daily returns from the U.S. stock market. In particular, we restrict the investment set to the Standard & Poor's 500 sectoral indices, plus the Bloomberg Barclay U.S. Aggregate Bond index. The latter is a composite indicator including Treasuries, government-related and corporate securities, MBS (agency fixed-rate pass-throughs), ABS and CMBS (agency and non-agency) with daily returns showing an R^2 of 0.89 with the related treasury only index.[3]

We do a horse race in the form of a recursive out-of-sample comparison among different investment strategies, including: buy and hold startin from equally weighted, equally weighted, mean-variance, minimum variance, risk-parity and hierarchical risk-parity.[4] For these approaches estimates of the

[3]We also carry out an additional evaluation on weekly data and with the Bloomberg US Treasury Index instead of the Bloomberg Barclay U.S. Aggregate Bond index, whose results are reported in Appendix C.

[4]See Section 3.3 for additional details.

parameters (mean, covariance or correlation matrix) are carried out both with expanding window sample estimates and through the D^2FM of Chapter 4.

We compare these methods to other Deep Learning Portfolio Optimization (DLPO) approaches. In particular, we specify different approximations of equation (6.3) before applying the direct policy gradient approach via equation (6.7). Following *Noguer and Srivastava* [24], in one of our specification we use the first moment only. However, we also use up to the second, third and fourth moments. Finally, we also compare against the Sharpe ratio objective proposed in *Zhang et al.* [23]. All of these deep learning approaches share the same set of possible deep learning architectures, however we select for each of them separately the best architecture within this shared set based on the performance achieved on the last 10% of the in-sample data. We provide as input either the raw historical returns or the state vectors estimated by the D^2FM. Overall, we compare a total of 20 strategies in a recursive out-of-sample exercise. Namely, starting from the beginning of the out-of-sample portion of the data, each of the model is provided with past data to then output asset allocation weights for the next period. This procedure is repeated until the end of the out-of-sample. We next provide additional details on the data, training and evaluation methodologies adopted.

6.2.1 Data and Training Details

The dataset comprises 3830 daily observations starting 3 January 2006 and ending on 15 March 2021. The investment set and the summary statistics are described in Table 6.1.

The out-of-sample starts the 1^{st} of January 2011 and runs until the end of the sample. We allow for daily rebalancing for all the compared approaches. Furthermore, we implement a daily re-estimation for the parameters of the otherwise static approaches: mean-variance, minimum-variance, risk-parity and hierarchical risk parity. Regarding deep learning methods, we re-estimate the model at the beginning of every year and validate the hyperparameters on the last 10% of the sample. With respect to such hyperparameters, we tune the

Asset	Standard Deviation	Mean	Skewness	Kurtosis
S&P 500 Information Technology Sector GICS Level 1 Index	1.45	0.06	-0.07*	9.92***
S&P 500 Energy Sector GICS Level 1 Index	1.90	0.02	-0.2***	14.68***
S&P 500 Financials Sector GICS Level 1 Index	2.09	0.03	0.32***	15.72***
S&P 500 Utilities Sector GICS Level 1 Index	1.24	0.02	0.41***	17.95***
S&P 500 Consumer Discretionary Sector GICS Level 1 Index	1.38	0.05	-0.14***	9.82***
S&P 500 Health Care Sector GICS Level 1 Index	1.12	0.04	-0.01	11.5***
S&P 500 Industrials Sector GICS Level 1 Index	1.41	0.04	-0.29***	9.08***
S&P 500 Consumer Staples Sector GICS Level 1 Index	0.94	0.03	0.06	14.77***
S&P 500 Materials Sector GICS Level 1 Index	1.60	0.04	-0.25***	8.45***
S&P 500 Communication Services Sector GICS Level 1 Index	1.32	0.03	0.23***	11.78***
S&P 500 Real Estate Sector GICS Level 1 Index	2.14	0.04	0.41***	17.03***
Bloomberg Barclays US Agg Total Return Value Unhedged USD	0.23	0.02	-0.36***	4.27***

Table 6.1: List of indices used with sample statistics computed on daily returns. For Skewness and Kurtosis, we conduct a statistical test of deviation from normality. Significance levels are: * for 10%, ** for 5% and *** for 1%.

number of lags on the input variables in $\{0, 5, 21, 63\}$, the structure of the neural networks both in terms of number of neurons (between 3 and 5) and type of layers (LSTM, fully connected *relu* or *tanh*), and the seed impacting parameters initialisation. This selection is repeated at each re-estimation step. Training is carried out using ADAM with default parameters; we stop the training if the validation loss does not improve for 3 consecutive epochs. Finally, the same validation approach is used for the hyperparameters of the D^2FM; this time the objects of validation are the number of lags on the same grid as above

and the encoder structure. In particular, the number of hidden units is in $\{40\text{-}20\text{-}10, 32\text{-}16\text{-}8, 20\text{-}10\text{-}5, 16\text{-}8\text{-}4, 20\text{-}4\}$, while the encoder is assumed to be fully connected with *tanh* and batch normalisation layers.

6.2.2 Evaluation Metrics

We use a number of evaluation metrics commonly adopted for the assessment of quantitative investment strategies. In particular, we evaluate the out-of-sample performance of the different approaches via the following metrics:

$$Sharpe\,ratio = \sqrt{252}\frac{E(r_p)}{Std(r_p)} \tag{6.8}$$

where $E(r_p)$ is computed using the out of sample average portfolio return, while $Std(r_p)$ is the out of sample standard deviation;

$$Sortino\,ratio = \sqrt{252}\frac{E(r_p)}{Std(r_p^-)} \tag{6.9}$$

where $Std(r_p^-)$ is the standard deviation computed only on negative returns;

$$maximum\,drawdown = \max_{0\leq t_1\leq t_2\leq T}(\frac{P_{t_1}^p - P_{t_2}^p}{P_{t_1}^p}) \tag{6.10}$$

where P_t^p is the actual value of the portfolio at time t equal t_1 and t_2;

$$turnover = \frac{252}{T}\sum_{t=1}^{T}\sum_{i=1}^{n}|w_{i,t} - \frac{(1+r_{i,t-1})w_{i,t-1}}{1+\sum_{i=1}^{n} r_{i,t-1}w_{i,t-1}}|. \tag{6.11}$$

6.2.3 Results and Discussion

We report the results of our exercise in Table 6.2.

By analysing Table 6.2, we note that all the strategies provide better risk adjusted returns in terms of Sharpe and Sortino ratios compared to an equally weighted allocation and the buy and hold strategy, the being constructed starting from an equally weighted portfolio and without rebalancing. The only exception is the mean-variance portfolio with the D^2FM prediction. In

Input Data	Method	Sharpe Ratio	Sortino Ratio	Maximum Drawdown	Turnover
	buy and hold	0.69	0.80	32.56	0.00
	equally weighted	0.66	0.77	33.71	1.28
D^2FM predictions	mean-variance	0.69	0.73	36.07	255.81
	minimum variance	1.43	1.75	8.02	0.59
	risk-parity	0.74	0.82	41.29	2.32
	hierarchical risk-parity	0.89	1.08	31.86	0.70
Sample average	mean-variance	0.74	0.88	31.21	10.57
	minimum variance	1.29	1.64	7.12	1.52
	risk-parity	0.77	0.91	33.40	1.09
	hierarchical risk-parity	0.89	1.10	32.09	0.34
D^2FM states	DLPO sharpe	0.94	1.05	16.33	89.91
	DLPO first moment	0.71	0.85	35.31	223.91
	DLPO first two moments	0.74	0.90	31.58	204.61
	DLPO first three moments	0.74	0.89	31.54	200.53
	DLPO first four moments	0.74	0.89	31.55	200.52
Plain inputs	DLPO sharpe	0.81	0.93	16.82	105.30
	DLPO first moment	0.77	0.85	28.48	150.52
	DLPO first two moments	0.79	0.90	28.35	142.24
	DLPO first three moments	1.04	1.41	21.47	114.00
	DLPO first four moments	1.06	1.44	20.84	118.77

Table 6.2: Recursive out-of-sample results of the different asset allocation strategies for the period 01/01/2011 to 15/03/2021. DLPO stands for Deep Learning Portfolio Optimisation methods. All the Deep Learning models are re-estimated yearly. The states and predictions of the D^2FM are updated daily via the Kalman Filter. Colours go from best blue to worst red.

particular, with the exception of the minimum variance strategy and the Deep Learning one with Sharpe ratio as objective function, all others show a reduction in their performance when inputs are provided from the D^{2}FM. Thus, indicating they do not combine well with such model. This is not suprising for the Deep Learning Portfolio Optimisation (DLPO) approaches involving higher moments, as at the current stage the D^{2}FM does not take into consideration such higher order moments.

However, the highest performance model is the minimum variance with the covariance matrix provided by the D^{2}FM. This approach provides the best performance both in terms of Sharpe and Sortino ratios. Namely, it outperforms its counterpart using the sample estimate of the covariance matrix (i.e., the second best approach) by 11% in terms of Sharpe ratio and 7% in terms of Sortino ratio, albeit sacrificing 13% in terms of maximum drawdown. What's more, its turnover is also among the lowest.

The next top performing approach is the Deep Learning based portfolio optimisation with four moments (DLPO first four moments) and the raw returns as input. This is the best performing method among the Deep Learning ones in terms of risk adjusted returns. Among the DL approaches that use a moment based loss function, it is also the best in terms of maximum drawdown.[5] Finally, with respect to the annual portfolio turnover we note that the DL based approaches make much more reallocations compared to standard financial approaches.

We continue our analysis in Table 6.3 where we report the results of a linear regression of the selected strategies excess returns against five factors coming from the financial literature on factor investing.[6] Our aim here is to check whether these strategies can be linearly replicated by standard financial factors, and whether, after correction for such factors, there is a significant alpha. Among the many factors present in the Financial Economics literature,

[5]This result is confirmed in the robustness check carried out in Appendix C.

[6]We download data for the factors from AQR website: https://www.aqr.com/Insights/Datasets/Betting-Against-Beta-Equity-Factors-Daily.

we select the most important ones; in particular, we choose the betting against beta (BAB) factor from *Frazzini and Pedersen* [35], the market (MKT), small minus big or size (SMB), and high minus low or value (HML) factors from the Fama–French three-factor model [31, 32, 33], and the up minus down or momentum (UMD) factor introduced by *Carhart* [34].

	equally weighted	minimum variance	DLPO first second third fourth moment	DLPO first moment	DLPO sharpe (DDFM states)
const	0.0001	0.0001*	0.0005**	0.0002	0.0003
	(0.0002)	(0.0001)	(0.0003)	(0.0003)	(0.0002)
BAB	0.4232***	0.0587***	0.1662***	0.4501***	0.1254***
	(0.0498)	(0.0101)	(0.0509)	(0.0567)	(0.0385)
MKT	0.1135***	0.0148***	0.0929***	0.1585***	0.0362**
	(0.0235)	(0.0048)	(0.0240)	(0.0268)	(0.0182)
SMB	0.2821***	0.0013	0.1004*	0.2091***	0.0467
	(0.0536)	(0.0109)	(0.0547)	(0.0610)	(0.0414)
HML	0.0951*	0.0165	0.0885	0.2219***	0.0051
	(0.0560)	(0.0114)	(0.0571)	(0.0637)	(0.0432)
UMD	-0.0942**	-0.0072	-0.0414	-0.0455	-0.0465
	(0.0434)	(0.0088)	(0.0443)	(0.0493)	(0.0335)
R-squared	0.0729	0.0307	0.0206	0.0718	0.0100

Table 6.3: OLS regression of selected strategies excess returns against financial factors. BAB stands for 'betting against beta factor', MKT for the 'market factor', SMB for 'small minus big', HML for 'high minus low', UMD for 'up minus down'. All the factors are taken from the AQR website. Significance levels are: * for 10%, ** for 5% and *** for 1%.

Regression results in Table 6.3 show that the only two factors that are significant in linearly explaining excess returns consistently among all the selected strategies are BAB and MKT. Furthermore, the only two strategies that generate a significant alpha over the out of sample are the minimum variance and the Deep Learning one with four moments. Additionally, only the alpha of the latter is significant at the 5% level. Nevertheless, once we correct for robust standard errors, all the regression parameters become insignificant. Moreover, the variance explained by those factors is below 10%, indicating the model is clearly misspecified.

6.3 Shapley Attributions

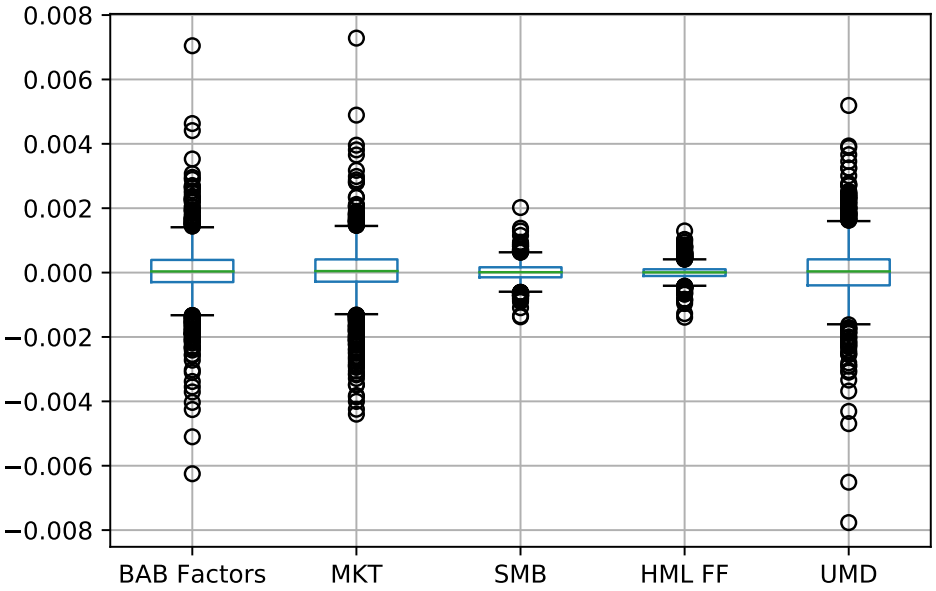
Moehle et al. [282] use Shapley Value to compute portfolio performance attribution with respect to given features. In this section we present a methodology to compute nonlinear factor exposure and portfolio performance attribution to financial factors. The methodology is based on Shapley values decomposition and the baseline proposed in Chapter 5.

Indeed, OLS regressions can only linearly attribute the performances of the strategies to each of the canonical financial factors. Hence, we go a step further and use multi-layer perceptrons to check whether the strategies can be better explained by the financial factors via the nonlinear formulations spanned by these models. To this purpose we validate different network structures on the data, including the linear model. However, we find that for most of the strategies the best model among all the ones tested on the data under analysis is the linear model. The only exceptions are the four moments DLPO and the DLPO Sharpe with states estimated by D^2FM as inputs. However, also for these the variances explained by linear and nonlinear models are very close to each other. Nonetheless, for the sake of completeness and as an illustrative example we decide to use Shapley values and our baseline approach introduced in Chapter 5 to decompose excess returns in terms of each of the risk factors for these two strategies. Namely, we estimate models of the following form

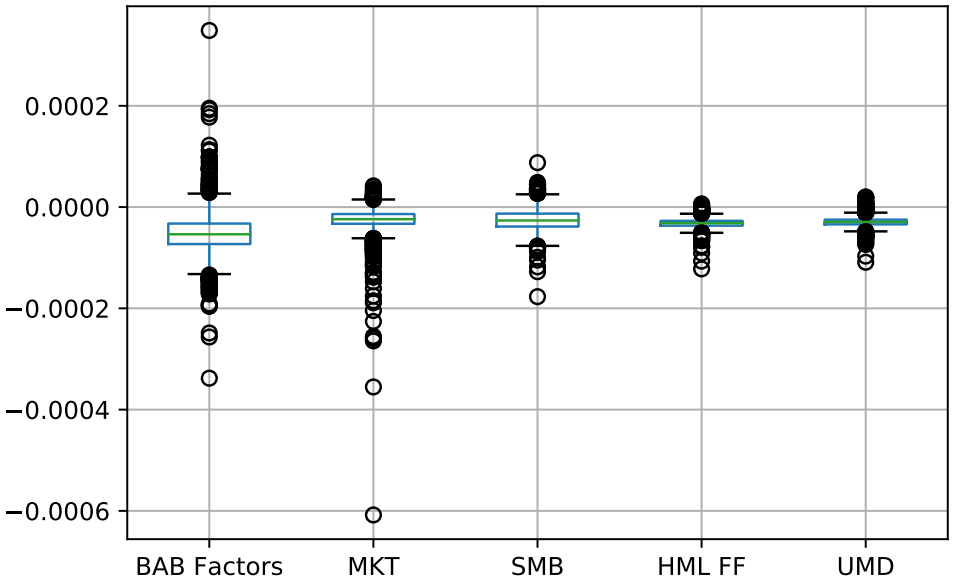
$$r^{pr}_{p,t} = G_{\boldsymbol{\theta}}(\boldsymbol{f}_t) + \epsilon_t, \tag{6.12}$$

with $r^{pr}_{p,t} = r_{p,t} - r_{f,t}$, that is the return of a selected portfolio (DLPO or the DLPO Sharpe) on top of the risk free rate; while $\boldsymbol{f}_t$ is the vector collecting the financial risk factors. Finally, $G_{\boldsymbol{\theta}}(\cdot)$ is the selected MLP and ϵ_t is the error term. All models are trained with a mean square error loss, which is proportional to the likelihood when residuals are i.i.d. Gaussian. Once the model has been estimated we compute its Shapley values. In particular, we set 0 as neutrality level for the output of the model, as we want to explain excess returns on top of the risk-free. Figures 6.1a and 6.1b display Shapley values computed

on the strategies whose validation exercise selects neural networks as the best performing model instead of the linear one.



(a) Four moments DLPO.



(b) DLPO sharpe (D^2FM states)

Figure 6.1: Box-plot of Shapley values with neutral baseline and neutrality value set to 0.

Figure 6.1a shows that the financial factors contribute half of the time positively and half of the time negatively to the excess returns generated by the 4 moments DLPO strategy. Whereas, Figure 6.1b indicates that for the DLPO with Sharpe loss and the D^2FM states the box-plots of HML and UMD are entirely below zero; therefore, they almost always contribute negatively to the excess returns generated by this strategy. In order to assess whether this finding is correct, we added to both strategies a short position on a portfolio, which is half invested in the High Minus Low (HML) factor and half invested in the Up Minus Down (UMD) factor. We found that the Sharpe ratio of the DLPO with Sharpe loss (and D^2FM states) improved by 10.1% compared to the 5.9% improvement of the four moments DLPO.

6.4 Discussion

In this chapter we have introduced an approach to combine portfolio choice theory with Deep Learning and Deep Reinforcement Learning. We have provided a possible, albeit exploratory, framework for the dynamic portfolio optimisation problem that takes into account higher order moments, and allows to compute nonlinear portfolio performance attribution and exposure to financial-economic factors. For the former, we used a simple gradient based direct deterministic policy only deep learning method without a *critic network*. We then also combined all the different methods analysed with the D^2FM of Chapter 4. With respect to the portfolio performance attribution, we show a way to compute this via Shapley values and the baseline introduced in Chapter 5.

Empirical analysis on U.S. market data shows the approach has potential. Nevertheless, these results also show that combining a classicial minimum variance approach with the D^2FM introduced in Chapter 4 delivers the best risk-adjusted performances. We think more advanced algorithms from Reinforcement and Deep Reinforcement Learning, such as the *actor-critic* Deep Deterministic Policy Gradient (DDPG) introduced in *Silver et al.* [206], could

benefit the framework. Furthermore, nongradient and hybrid optimisation tools could also better the performance given the complexity of the objective derived especially when including higher order moments.

Chapter 7

Conclusion

This chapter summarises the main findings of this PhD thesis, sets out the limitations of the studies, and outlines future work.

7.1 Summary and Assessment

In Chapter 4 Deep Dynamic Factor Models (D^2FM) are introduced; this novel class of models merges the deep learning (DL) literature on autoencoders with that of the Econometrics on Dynamic Factor Models (DFMs). The approach allows to estimate state space models with factor structure. Those are useful to represent time series data generated from unobservable stochastic dynamic common and idiosyncratic components. The framework accounts for mixed frequencies and missing data. The empirical applications on synthetic and macroeconomic *real-time* data show the potential and generality of such a methodology. In particular, we demonstrate in a Monte Carlo study the ability of the D^2FM to effectively capture nonlinearities in the data generating process (DGP), while still being able to show competing performances to the standard DFM when the DGP is linear. The superior performance is confirmed on real time macroeconomic data, where the D^2FM shows improved performances in forecasting and nowcasting US real GDP growth over the out of sample period between beginning of 2006 and mid 2020. Moreover, it maintains some intepretability aspects of standard DFMs. That is, while forecasting in real-time

with non-synchronously published data[1] it is useful to attribute the unexpected data revision to the specific new piece of information released. This can be done in the proposed framework using the same methodology adopted in DFMs, where the so called *news* and *impact* are computed as discussed in *Bańbura and Modugno* [39]. Moreover, it is possible to construct synthetic indicators and interpret the estimated latent states as features of the business cycles. A note of caution is however needed in regard to the factors estimated via the D^{2}FM, as we did not formally analyse their statistical properties, neither we discussed ways to achieve their identification. Those topics have been well studied for DFMs in the econometric literature and require further work to be potentially extended to the D^{2}FM.

In Chapter 5 a novel baseline for feature attribution methods and Shapley values is proposed. The baseline is chosen according to the concept of *fairness* and that of *neutrality*. A neutrality value is a parameter selected by decision-makers, it represents the point at which their choices are determined by the model predictions being either above or below it. Thus, the proposed baseline is set based on a parameter that depends on the actual use of the model. This procedure stands in contrast to how other baselines are set, i.e. without accounting for how the model is used. The chapter introduces also the concept of fairness which is needed to represent features fairly in their realisation space, and given their relation to the model. This allows to take into consideration distributional differences among the features to avoid selecting neutral baselines that could deliver to distorted attributions because of such differences. Various examples and empirical evidence from synthetic data and a dataset derived from the financial domain corroborate these arguments. Nevertheless, the approach proposed here has both computational and applicability limitations which deserve further research. Moreover, it misses a comparison with more recent methods such as the ones based on counter-factual explanation [see 283, 284, for example].

[1] A standard characteristic of real-time economic data, which features publication lag and data revision.

Lastly, in Chapter 6 it is discussed and evaluated an approach to combine portfolio choice theory with Deep Learning and Deep Reinforcement Learning. The framework allows for dynamic portfolio optimisation with higher order moments, and nonlinear portfolio performance attribution to financial-economic risk factors. For the latter, we adopt Shapley values and the baseline introduced in Chapter 5 to decompose the premium of an investment strategy with respect to economic and financial risk factors in a manner that need not be constrained to a linear formulation. Overall, the findings of Chapter 6 support the potential of hybrid strategies merging financial theory with machine learning for investing. Nevertheless, further research and experiments are needed to effectively adopt the methods discussed.

7.2 Future Work

In the future we would like to analyse more formally the statistical properties of the estimation framework of the D^{2}FM, and look for potential improvements. On the applicative aspects we think it could be of interest to extend the approach to test whether alternative data (e.g., text data, satellite images) can better the empirical performance of the model [see 107, 51, for example]. Moreover, applications to analyse the conditional distribution of GDP growth by combining the model with a quantile approach as in *Adrian et al.* [285] and in *Reichlin et al.* [286] are other areas to explore.[2] Additionally, imposing a penalty in the loss to force identification of the factors is another interesting area of research. In particular, in *Bardes et al.* [287] a new method called VICReg (Variance-Invariance-Covariance Regularization) is introduced. The method specifies a loss function which takes into consideration the distribution of the latent states by forcing them to have zero covariance, while maintaining a variance above a given threshold. The zero covariance term is aimed to prevent a so called *informational collapse* in which the variables would be highly correlated. This idea is borrowed from the Barlow Twins method of

[2] In this regard it could be interesting to consider also the framework proposed in *Chen et al.* [278].

Zbontar et al. [288], and we think it is an interesting approach for the inclusion of a penalisation term to force identification of the factors. In the future we would like to work on this aspect as well, including the potential of monotonic and sign restrictions. Finally, with respect to the hyperparameter selection strategy used, and being the model able to handle missing data, it can be of interest to experiment different estimation methods of the generalisation errors based on which hyperparameters are selected including those adopting an artificial deletion of the observations, such as the one proposed by *Pellegrino* [289], but also others based on pre-filtering of the models via statistical tests as in *Bergmeir et al.* [115].

The new baseline proposed in Chapter 5 to compute feature attributions has both computational and applicability limitations that require further research. In particular, and based on our findings, the approach could be extended to learn baselines by specifying a loss function that takes into account the concepts of *neutrality* and *fairness*. A first attempt in this direction has been given via the use of convex optimisation when features are Gaussian. Nevertheless, further relaxations and generalisations are unequivocally needed to extend the approach to more complex and general problems. Moreover, a comparison of the approach with more recent methods from the literature, such as the ones based on counter-factual explanation [see 283, 284, for example] is needed.

The approach discussed in Chapter 6 necessitates more research on the algorithmic design. Indeed, more advanced and hybrid optimisation algorithms, and (Deep) Reinforcement Learning approaches such as the one discussed in *Silver et al.* [206], could benefit the framework. Additional theoretical and empirical analysis would also be beneficial to the work. On the empirical aspect, performing multiple evaluations on different datasets and including additional strategies such as the ones discussed in the literature review chapter would only serve to enhance the assessment of the methodologies. Also, evaluations in synthetic environments with well known optimal allocation strategies could shed light on whether the approach is able to recover the otherwise known

best investment actions. With respect to the factor exposure, different model specifications including additional covariates could be considered and evaluated. Morevoer, similarly to *Nakagawa et al.* [136, 138] interpretable versions of the models could be constructed and compared to ex-post explanation strategy. We leave these areas of enquiry to future research.

Furthermore, in Appendix A it is illustrated a way to compute a full decomposition of the statistical latent factors learned by the model introduced in Chapter 4, and more in general of latent states of state space models estimated via deep learning methods. Indeed, in these cases latent states are computed as the output of a map that can be additively decomposed. These latent states are then updated via the use of statistical filtering (e.g., the Kalman filter). In doing so, the approach merges Shapley values with the interpretability of the filtering process that allows for online updates of the attributions initially computed via the Shapley values. Notwithstanding the methodology is theoretically appealing, it misses a proper assessment which is deferred to future work.

Finally, the combination of approaches and empirical results discussed in this PhD thesis could allow for a unique framework to make systematic and macroeconomic informed investment decisions, whose performance can then be explained in relation to fundamental risk factors from the financial economic literature [see 31, 32, 33, 34, 35, for example]. Notwithstanding here we analyse some of these aspects, more work is unequivocally needed to effectively combine the methods discussed, including their possible extensions and improvements, given their limitations.

Appendix A

Appendix to Chapter 4

A.1 Deep Learning and State-Space: Shapley Meets Filters

In this appendix section we discuss how to extend feature attribution methods to state space models estimated via deep learning techniques. The time series under analysis are assumed to be continuous. The key idea of the approach is to consider the latent states as the output of an encoder network, while the decoder represents the emission equation of a state space model. In this way Shapley values are applied only at each re-estimation step on the encoder network; when in production, the latent states and their explanations are updated via the use of filters. This allows for a considerable computational gain in a well-grounded theoretical setting, given that the computational cost of exact Shapley values is exponential, while that of filters is polynomial. Let us begin by reviewing common filtering techniques in State Space Models, then move to a discussion of the core contribution, which consists of merging Shapley values with filters.

A.1.1 State Space Models and Filtering

State space models provide a highly flexible framework to model time series data. These models assume that the driving forces of the development over time of the observable series need to be found in some unobservable components. Once

the parameters of the model have been estimated, updates of the unobservable components given the current observation are carried out in real time, as these observations are made available to the model. This process is called filtering. In this section we review common approaches to the optimal filtering problem, following *Haykin* [290], and *Durbin and Koopman* [291] who we refer for further details.

Since our focus is on Gaussian State Space models (GSSMs) with additive noise, we start by formally introducing such models. Extensions to models with non-Gaussian errors are possible through the use of appropriate filters [see 290, 291, for details].

The observable or measurement equation is given by:

$$\mathbf{y}_t = F(\boldsymbol{a}_t) + \boldsymbol{v}_t \quad \boldsymbol{v}_t \sim \mathcal{N}(0, \boldsymbol{R}_t), \tag{A.1}$$

with $\boldsymbol{R}_t$ being a diagonal covariance matrix and $F(\cdot)$ a generic function. The state equation is given by

$$\boldsymbol{a}_t = B(L)\boldsymbol{a}_t + \boldsymbol{w}_t \quad \boldsymbol{w}_t \sim \mathcal{N}(0, \boldsymbol{Q}_t) \tag{A.2}$$

where $B(L)$ is a generic autoregressive lag polynomial, while $\boldsymbol{Q}_t$ is a diagonal covariance matrix. Additionally, $\boldsymbol{w}_t$ is assumed to be uncorrelated with $\boldsymbol{v}_t$.

A.1.2 The Kalman Filter

The celebrated *Kalman filter* [292] provides a recursive solution to the linear optimal filtering problem. In particular, given the following state-space model:

$$\mathbf{y}_t = F\boldsymbol{a}_t + \boldsymbol{v}_t \quad \boldsymbol{v}_t \sim \mathcal{N}(0, \boldsymbol{R}_t) \tag{A.3}$$

$$\boldsymbol{a}_t = B\boldsymbol{a}_{t-1} + \boldsymbol{w}_t \quad \boldsymbol{w}_t \sim \mathcal{N}(0, \boldsymbol{Q}_t) \tag{A.4}$$

with F and B being matrices, while $\boldsymbol{v}_t$, $\boldsymbol{w}_t$ and their covariances follow the assumptions stated above. The Kalman filter works as follows [290]:

1. Initialise $\widehat{\boldsymbol{a}}_0 = E[\boldsymbol{a}_0]$, $\boldsymbol{P}_0 = E[(\boldsymbol{a}_0 - E[\boldsymbol{a}_0])(\boldsymbol{a}_0 - E[\boldsymbol{a}_0])^\top]$

2. Computation for $t = 1, 2, \ldots$

   (a) State estimate propagation $\widehat{\boldsymbol{a}}_{t|t-1} = B\widehat{\boldsymbol{a}}_{t-1|t-1}$

   (b) Error covariance propagation $\boldsymbol{P}_{t|t-1} = B\boldsymbol{P}_{t-1|t-1}B^\top + \boldsymbol{Q}_{t-1}$

   (c) Kalman gain matrix $K_t = \boldsymbol{P}_{t|t-1}F^\top[F\boldsymbol{P}_{t|t-1}F^\top + \boldsymbol{R}_t]^{-1}$

   (d) State estimate update $\widehat{\boldsymbol{a}}_{t|t} = \widehat{\boldsymbol{a}}_{t|t-1} + K_t(\mathbf{y}_{t|t} - F\widehat{\boldsymbol{a}}_{t|t-1})$

   (e) Error covariance update $\boldsymbol{P}_{t|t} = (I - K_tF)\boldsymbol{P}_{t|t-1}$.

A.1.3 Nonlinear Filters

In order to extend the Kalman filter to state space models with nonlinear transition and/or dynamics, we need to resort to approximation methods: *Extended Kalman Filters (EKFs)*, *Unscented Kalman Filters (UKFs)* and *Particle Filters (PFs)*. We focus on the first two, as we restrict our analysis to GSSMs.

The EKF [293, 294] consists in carrying out a linearisation of the system by means of a Taylor expansion around the most recent state. Since in a non linear GSSM both F and B can be nonlinear, when performing the filtering step we replace those matrices with an appropriate approximation, e.g., with a first order Taylor expansion $\widetilde{F}_t = \frac{\partial F(\boldsymbol{a})}{\partial \boldsymbol{a}}|_{\boldsymbol{a}=\boldsymbol{a}_{t|t-1}}$ and $\widetilde{B}_t = \frac{\partial B(\boldsymbol{a})}{\partial \boldsymbol{a}}|_{\boldsymbol{a}=\boldsymbol{a}_{t|t-1}}$. All the rest works as in the standard Kalman filter.

The EKF achieves only a first-order accuracy; better approximation can be achieved using a second order expansion but this would require additional computational cost to calculate the Hessian Matrix. By way of contrast, the UKF [295, 296, 297, 298, 299], which does not require computing derivatives, has the same computational cost as the EKF, but delivers a third-order accuracy for Gaussian inputs for all nonlinearities, while maintaining at least a second-order accuracy for non-Gaussian inputs [290]. The idea behind the UKF is to select appropriate sample points and propagate them trough the nonlinear system, which allows to compute the posterior mean and covariance for any

nonlinearity, with accuracy up to the aforementioned orders. This process recalls the *Gaussian quadrature*; indeed, the relation between the two is shown in *Ito and Xiong* [300].

A.1.4 Interpretability of the Filters

In addition to their extensively studied theoretical properties regarding the updates of the latent states given the observables, filters allow for easy interpretation, as the contribution of each observable to each of the latent states is automatically computed during the update step by leveraging on the state-space relations and the algebraic properties of the filtering.

Certainly, if we define the *news* [93] vector as $\mathbf{y}_{t+1|t+1} - F\widehat{\boldsymbol{a}}_{t+1|t}$, i.e., the unexpected variation in the observable, we can directly attribute each of the element of such a vector to the latent states through the structure of the Kalman Gain matrix (or its approximations in the nonlinear case). In fact, the variation in the latent state, k, attributed to observable, i, is given by:

$$K_{t+1}^{k,i}(\mathbf{y}_{t+1|t+1} - F\widehat{\boldsymbol{a}}_{t+1|t})^i, \tag{A.5}$$

where $K_{t+1}^{k,i}$ is the k,i element of the matrix and $(\mathbf{y}_{t+1|t+1} - F\widehat{\boldsymbol{a}}_{t+1|t})^i$ is the $i-th$ row of such news vector. For the *EKF* and the *UKF*, we need to replace $K_{t+1}^{k,i}$ with its approximations [see 298, for example].

A.1.5 Merging Shapley Values with Filters

When a state space of the form A.1 - A.2 is estimated via a deep learning approach, as in the model of Chapter 4, to fully explain the latent states with respect to each of the observables, we need to merge a feature attribution method with a filter. In particular, when such state-space is augmented with an encoder for the latent states (i.e., $\boldsymbol{a}_t = G(\mathbf{y}_t)$), Shapley values could be applied to explain $G(\cdot)$; and then the properties of the chosen filter could be used to attribute subsequent variations in the unobservable components.

Why we need the merge? Let us assume we are at time $t < T$ and a linear GSSM has been estimated with all the data up to this point in time. Now

we will be rolling the model in production for the next $\{t+1,\ldots,T\}$ periods. Then, the first update step at $t+1$ is given by the following equation:

$$\widehat{\boldsymbol{a}}_{t+1|t+1} = \underbrace{\widehat{\boldsymbol{a}}_{t+1|t}}_{Init} + \underbrace{K_{t+1}(\mathbf{y}_{t+1|t+1} - F\widehat{\boldsymbol{a}}_{t+1|t})}_{Upd} \tag{A.6}$$

where $\widehat{\boldsymbol{a}}_{t+1|t} = B(L)G(\mathbf{y}_t)$, with $B(L)$ being a lag-polynomial and G being a neural network. When using filters we can attribute only the update part (Upd) to each of the new observations coming into the model using equation (A.5), while the initial part ($Init$) of equation (A.6) remains unattributed and hence unexplained. Therefore, we need a methodology to decompose it.

How we do the merge? Fortunately, Shapley values and its approximations can help us to additively decompose the $Init$ part of equation (A.6). If we define with $FAD_{k,i}(\widehat{\boldsymbol{a}}_{t+1})$ the feature attribution decomposition of the latent state with index k in $\widehat{\boldsymbol{a}}_{t+1}$ with respect to observable i, then at time $t+1$, we have:

$$FAD_{k,i}(\widehat{\boldsymbol{a}}_{t+1}) = B^{k,:}\, Shap_{\boldsymbol{b}}(G(\mathbf{y}_t))_{:,i} + K^{k,i}_{t+1}(\mathbf{y}_{t+1|t+1} - F\widehat{\boldsymbol{a}}_{t+1|t})^i, \tag{A.7}$$

where $B^{k,:}$ represents the $k-th$ row of the matrix of the dynamics, assuming a linear autoregressive process in $B(L)$, while $Shap_{\boldsymbol{b}}(G(\mathbf{y}_t))_{:,i}$ is the Shapley decomposition vector of all the latent states with respect to the feature (observable) i at time t, and at baseline $\boldsymbol{b}$. Nevertheless, we still need to define an appropriate baseline. A way to do this is to define a neutrality level on the latent states and then use the methodology of Chapter 5 to find the baseline value on the observable space. A natural choice for this neutrality level could be the $0's$ vector. It is noteworthy that this zero vector differs from the zero baseline, as it is defined on the latent states, not in the observable space. This choice is reasonable, as intuitively, when the unobservable components are at this value, they do not have any influence on the observables. Furthermore, these unobservable components can be seen as neurons from the neural network

perspective, and the use of the zero value to model absence can be justified for neurons because it signals when the neuron is dead [301].

Once the $t+1$ attribution has been computed, next period decompositions can be recursively calculated from the past ones, thus requiring only a propagation of the past feature attribution towards the system dynamics, and the *Upd* term of equation (A.6) to be computed. The high computational cost for calculating the Shapley values is therefore sustained only at each re-estimation step. Indeed, at $t+h$ with $h>1$, the feature attribution decomposition of latent state k with respect to observable i is given by:

$$FAD_{k,i}(\widehat{\boldsymbol{a}}_{t+h}) = B^{k,:}\, FAD_{:,i}(\widehat{\boldsymbol{a}}_{t+h-1}) + K^{k,i}_{t+h}(\mathbf{y}_{t+h|t+h} - F\widehat{\boldsymbol{a}}_{t+h|t+h-1})^i, \quad \text{(A.8)}$$

where here again in $B^{k,:}$ we assume a linear autoregressive process for the unobservable dynamics. In case the state dynamics in equation (A.2) are nonlinear and cannot be well approximated by a linearisation, then one can use again Shapley values to propagate the feature attribution into the dynamic equations. Importantly, even if Shapley based attributions would need to be used again for the dynamics, the problem has a much lower complexity, as usually the shared states (aka common factors) in the dynamic equations are of a lower dimension compared to the observable cardinality. In the case of the D^2FM presented in the empirical part of this thesis in Chapter 4, being the system linear in the dynamics, a matrix multiplication as in equation (A.8) is all we need.

A.2 Data Appendix Macroeconomic Application

Tables A.1-A.4 reports the list of variables in the dataset. The transformation codes (Tcode) in the tables refer to how the variables are transformed to archive stationary. With x_t, as a raw series, the transformations adopted are:

$$z_t = \begin{cases} x_t & \text{if Tcode} = 1 \\ (1-L)\, x_t & \text{if Tcode} = 2 \\ (1-L)(1-L^{12})\, x_t & \text{if Tcode} = 3 \\ \log x_t & \text{if Tcode} = 4 \\ (1-L)\log x_t & \text{if Tcode} = 5 \\ (1-L)(1-L^{12})\log x_t & \text{if Tcode} = 6 \end{cases}$$

Table A.1: Dataset (I)

N	Code	Descriptions	Source	Tcode	Freq	McCracken gr	Publication delay
1	RPI	Real Personal Income	FRED	5	M	1	30
2	W875RX1	RPI ex. Transfers	FRED	5	M	1	30
3	INDPRO	IP Index	FRED	5	M	1	14
4	IPFPNSS	IP: Final Products and Supplies	FRED	5	M	1	14
5	IPFINAL	IP: Final Products	FRED	5	M	1	14
6	IPCONGD	IP: Consumer Goods	FRED	5	M	1	14
7	IPDCONGD	IP: Durable Consumer Goods	FRED	5	M	1	14
8	IPNCONGD	IP: Nondurable Consumer Goods	FRED	5	M	1	14
9	IPBUSEQ	IP: Business Equipment	FRED	5	M	1	14
10	IPMAT	IP: Materials	FRED	5	M	1	14
11	IPDMAT	IP: Durable Materials	FRED	5	M	1	14
12	IPNMAT	IP: Nondurable Materials	FRED	5	M	1	14
13	IPMANSICS	IP: Manufacturing	FRED	5	M	1	14
14	IPB51222S	IP: Residential Utilities	FRED	5	M	1	14
15	IPFUELS	IP: Fuels	FRED	5	M	1	14
16	CLF16OV	Civilian Labor Force	FRED	5	M	2	7
17	CE16OV	Civilian Employment	FRED	5	M	2	7
18	UNRATE	Civilian Unemployment Rate	FRED	2	M	2	7
19	UEMPMEAN	Average Duration of Unemployment	FRED	2	M	2	7
20	UEMPLT5	Civilians Unemployed <5 Weeks	FRED	5	M	2	7
21	UEMP5TO14	Civilians Unemployed 5-14 Weeks	FRED	5	M	2	7
22	UEMP15OV	Civilians Unemployed >15 Weeks	FRED	5	M	2	7
23	UEMP15T26	Civilians Unemployed 15-26 Weeks	FRED	5	M	2	7
24	UEMP27OV	Civilians Unemployed >27 Weeks	FRED	5	M	2	7
25	PAYEMS	All Employees: Total nonfarm	FRED	5	M	2	7
26	USGOOD	All Employees: Goods-Producing	FRED	5	M	2	7
27	CES1021000001	All Employees: Mining and Logging	FRED	5	M	2	7
28	USCONS	All Employees: Construction	FRED	5	M	2	7
29	MANEMP	All Employees: Manufacturing	FRED	5	M	2	7
30	DMANEMP	All Employees: Durable goods	FRED	5	M	2	7
31	NDMANEMP	All Employees: Nondurable goods	FRED	5	M	2	7
32	SRVPRD	All Employees: Service Industries	FRED	5	M	2	7
33	USTPU	All Employees: TT&U	FRED	5	M	2	7

Table A.2: Dataset (II)

N	Code	Descriptions	Source	Tcode	Freq	McCracken gr	Publication delay
34	USWTRADE	All Employees: Wholesale Trade	FRED	5	M	2	7
35	USTRADE	All Employees: Retail Trade	FRED	5	M	2	7
36	USFIRE	All Employees: Financial Activities	FRED	5	M	2	7
37	USGOVT	All Employees: Government	FRED	5	M	2	7
38	CES0600000007	Hours: Goods-Producing	FRED	1	M	2	7
39	AWOTMAN	Overtime Hours: Manufacturing	FRED	2	M	2	7
40	AWHMAN	Hours: Manufacturing	FRED	1	M	2	7
41	CES0600000008	Ave. Hourly Earnings: Goods	FRED	6	M	2	7
42	CES2000000008	Ave. Hourly Earnings: Construction	FRED	6	M	2	7
43	CES3000000008	Ave. Hourly Earnings: Manufacturing	FRED	6	M	2	7
44	HOUST	Starts: Total	FRED	4	M	3	20
45	HOUSTNE	Starts: Northeast	FRED	4	M	3	20
46	HOUSTMW	Starts: Midwest	FRED	4	M	3	20
47	HOUSTS	Starts: South	FRED	4	M	3	20
48	HOUSTW	Starts: West	FRED	4	M	3	20
49	PERMIT	Permits	FRED	4	M	3	20
50	PERMITNE	Permits: Northeast	FRED	4	M	3	20
51	PERMITMW	Permits: Midwest	FRED	4	M	3	20
52	PERMITS	Permits: South	FRED	4	M	3	20
53	PERMITW	Permits: West	FRED	4	M	3	20
54	DPCERA3M086SBEA	Real PCE	FRED	5	M	4	30
55	CMRMTSPL	Real M&T Sales	FRED	5	M	4	35
56	RETAIL	Retail and Food Services Sales	FRED	5	M	4	30
57	ACOGNO	Orders: Consumer Goods	FRED	5	M	4	35
58	ANDENO	Orders: Nondefense Capital Goods	FRED	5	M	4	35
59	AMDMUO	Unfilled Orders: Durable Goods	FRED	5	M	4	35
60	BUSINV	Total Business Inventories	FRED	5	M	4	35
61	ISRATIO	Inventories to Sales Ratio	FRED	2	M	4	35
62	UMCSENT	Consumer Sentiment Index	FRED	2	M	4	-3
63	M1SL	M1 Money Stock	FRED	6	M	5	14
64	M2SL	M2 Money Stock	FRED	6	M	5	14
65	M3SL	MABMM301USM189S in FRED, M3 for the United States	FRED	6	M	5	14
66	M2REAL	Real M2 Money Stock	FRED	5	M	5	14

Table A.3: Dataset (III)

N	Code	Descriptions	Source	Tcode	Freq	McCracken gr	Publication delay
67	AMBSL	St. Louis Adjusted Monetary Base	FRED	6	M	5	14
68	TOTRESNS	Total Reserves	FRED	6	M	5	36
69	NONBORRES	Nonborrowed Reserves	FRED	0	M	5	36
70	BUSLOANS	Commercial and Industrial Loans	FRED	6	M	5	20
71	REALLN	Real Estate Loans	FRED	1	M	5	20
72	NONREVSL	Total Nonrevolving Credit	FRED	6	M	5	14
73	MZMSL	MZM Money Stock	FRED	6	M	5	14
74	DTCOLNVHFNM	Consumer Motor Vehicle Loans	FRED	6	M	5	14
75	DTCTHFNM	Total Consumer Loans and Leases	FRED	6	M	5	14
76	INVEST	Securities in Bank Credit	FRED	6	M	5	14
77	FEDFUNDS	Effective Federal Funds Rate	FRED	2	M	6	-1
78	CP3M	3-Month AA Comm. Paper Rate	FRED	2	M	6	0
79	TB3MS	3-Month T-bill	FRED	2	M	6	0
80	TB6MS	6-Month T-bill	FRED	2	M	6	0
81	GS1	1-Year T-bond	FRED	2	M	6	0
82	GS5	5-Year T-bond	FRED	2	M	6	0
83	GS10	10-Year T-bond	FRED	2	M	6	0
84	AAA	Aaa Corporate Bond Yield	FRED	2	M	6	2
85	BAA	Baa Corporate Bond Yield	FRED	2	M	6	2
86	TB3SMFFM	3 Mo. - FFR spread	FRED	1	M	6	2
87	TB6SMFFM	6 Mo. - FFR spread	FRED	1	M	6	2
88	T1YFFM	1 yr. - FFR spread	FRED	1	M	6	2
89	T5YFFM	5 yr. - FFR spread	FRED	1	M	6	2
90	T10YFFM	10 yr. - FFR spread	FRED	1	M	6	0
91	AAAFFM	Aaa - FFR spread	FRED	1	M	6	0
92	BAAFFM	Baa - FFR spread	FRED	1	M	6	0
93	TWEXMMTH	Trade Weighted U.S. FX Rate	FRED	5	M	6	2
94	EXSZUS	Switzerland / U.S. FX Rate	FRED	5	M	6	2
95	EXJPUS	Japan / U.S. FX Rate	FRED	5	M	6	2
96	EXUSUK	U.S. / U.K. FX Rate	FRED	5	M	6	2
97	EXCAUS	Canada / U.S. FX Rate	FRED	5	M	6	2
98	PPIFGS	PPI: Finished Goods	FRED	6	M	7	16
99	PPIFCG	PPI: Finished Consumer Goods	FRED	6	M	7	16

Table A.4: Dataset (IV)

N	Code	Descriptions	Source	Tcode	Freq	McCracken gr	Publication delay
100	PPIITM	PPI: Intermediate Materials	FRED	6	M	7	16
101	PPICRM	PPI: Crude Materials	FRED	6	M	7	16
102	oilprice	Crude Oil Prices: WTI	HAVER	6	M	7	0
103	PPICMM	PPI: Commodities	FRED	6	M	7	16
104	CPIAUCSL	CPI: All Items	FRED	6	M	7	16
105	CPIAPPSL	CPI: Apparel	FRED	6	M	7	16
106	CPITRNSL	CPI: Transportation	FRED	6	M	7	16
107	CPIMEDSL	CPI: Medical Care	FRED	6	M	7	16
108	CUSR0000SAC	CPI: Commodities	FRED	6	M	7	16
109	CUUR0000SAD	CPI: Durables	FRED	6	M	7	16
110	CUSR0000SAS	CPI: Services	FRED	6	M	7	16
111	CPIULFSL	CPI: All Items Less Food	FRED	6	M	7	16
112	CUUR0000SA0L2	CPI: All items less shelter	FRED	6	M	7	16
113	CUSR0000SA0L5	CPI: All items less medical care	FRED	6	M	7	16
114	PCEPI	PCE: Chain-type Price Index	FRED	6	M	7	30
115	DDURRG3M086SBEA	PCE: Durable goods	FRED	6	M	7	30
116	DNDGRG3M086SBEA	PCE: Nondurable goods	FRED	6	M	7	30
117	DSERRG3M086SBEA	PCE: Services	FRED	6	M	7	30
118	IPMAN	Industrial Production: Manufacturing	FRED	1	M	1	14
119	MCUMFN	Capacity Utilization: Manufacturing	FRED	2	M	1	14
120	TCU	Capacity Utilization: Total Industry	FRED	2	M	1	14
121	M0684AUSM343SNBR	Manufacturers' Index of New Orders of Durable Goods	FRED	1	M	4	34
122	M0504AUSM343SNBR	Manufacturers' Inventories, Total for United States	FRED	1	M	4	34
123	DGORDER	Manufacturers' New Orders: Durable Goods	FRED	5	M	4	34
124	CPFFM	3-Month Commercial Paper Minus Federal Funds Rate	FRED	1	M	6	0
125	PCUOMFGOMFG	Producer Price Index by Industry: Total Manufacturing	FRED	1	M	7	15
126	ISMC	ISM Composite Index	HAVER	1	M	1	3
127	NAPMVDI	ISM Manufacturing: Supply Index	HAVER	1	M	1	3
128	SP500E	Standard & poor 500: Price Index	HAVER	5	M	8	0
129	SDY5COMM	Standard & poor 500: Dividend Yield	HAVER	2	M	8	0
130	SPE5COOM	Standard & poor 500: Price/Earnings Ratio	HAVER	5	M	8	0
131	GDPC1	Gross Domestic Product	FRED	5	Q	1	30

Appendix B

Appendix to Chapter 5

B.1 Proofs of Propositions 1 and 2

In this section we provide proofs of Proposition 1 and 2. For the convenience of the reader, some relevant assumptions and definitions of the paper are reviewed.

Assumptions:

A1. All activation functions are monotonic and continuous.

A2. All marginals cumulative distribution functions (CDFs) of the joint CDF of the input features are bijective and continuous.

Definition B.1.1 (Neutrality Value)**.** *Given a model prediction $\hat{y}$ and a decision maker, we say that the value α is neutral if the decision maker's choice is determined by the value of $\hat{y}$ being either below or above α.*

Definition B.1.2 (Space of Fair Baselines)**.** *Consider a dataset in $\mathbb{R}^k$, $k \geq 1$, generated by a distribution. A vector $\mathbf{x} \in \mathbb{R}^k$ is in the space of fair baselines (hence called a fair baseline) for a monotonic model $G_{\boldsymbol{\theta}}(\cdot)$ if*

$$\tilde{B} = \{\mathbf{x}^p \in \mathbb{R}^k : x_j^p = C_j^{-1}(\mathbb{1}_{\theta_j>0} \cdot p + \mathbb{1}_{\theta_j<0} \cdot (1-p)), \quad p \in [0,1],\, j = 1,2,\ldots,k\}, \tag{B.1}$$

where C_j^{-1}s are inverse marginal CDFs.

Proposition B.1.1 (Existence of a Neutral Fair Baseline for SLPs)**.** *Given an SLP $G_{\boldsymbol{\theta}}(\cdot)$ satisfying **A1**, a dataset satisfying **A2**, and a neutrality value*

α in the image of $G_{\boldsymbol{\theta}}(\cdot)$, then there exists at least a fair baseline $\mathbf{x}$ such that $G_{\boldsymbol{\theta}}(\mathbf{x}) = \alpha$.

Proof. We need to prove that $a \in G_{\boldsymbol{\theta}}(\tilde{B})$, where $G_{\boldsymbol{\theta}}(\tilde{B})$ is the image of $\tilde{B}$ under $G_{\boldsymbol{\theta}}$. Suppose that I is the image of the SLP. We show that $I \subseteq G_{\boldsymbol{\theta}}(\tilde{B})$, which proves the result, since $\alpha \in I$.

We start by showing that $\inf G_{\boldsymbol{\theta}}(\tilde{B}) \leq \inf I$ and that $\sup G_{\boldsymbol{\theta}}(\tilde{B}) \geq \sup I$. Consider vector $\mathbf{x}^0 \in \tilde{B}$ defined by $\mathbf{x}^0 = \{x_j^0 = C_j^{-1}(\mathbb{1}_{\theta_j<0}),\ \text{for all } j = 1, 2, \ldots, k\}$. So elements of $\mathbf{x}^0$ are the smallest possible if the coefficients are positive, and the largest possible when they are negative. From Assumption **A1**, it follows that $G_{\boldsymbol{\theta}}(\mathbf{x}^0)$ is the smallest value that the SLP can take. Hence, $G_{\boldsymbol{\theta}}(\mathbf{x}^0) \leq \inf I$.

Let us now take the vector $\mathbf{x}^1 \in \tilde{B}$ which is defined by: $\mathbf{x}^1 = \{x_j^1 = C_j^{-1}(\mathbb{1}_{\theta_j>0}),\ \text{for all } j = 1, 2, \ldots, k\}$. So elements of $\mathbf{x}^1$ are the largest possible if the coefficients are positive, and the smallest possible when they are negative. From assumption **A1**, it follows that $G_{\boldsymbol{\theta}}(\mathbf{x}^1)$ is the largest value that the SLP can take. Hence, $G_{\boldsymbol{\theta}}(\mathbf{x}^1) \geq \sup I$.

Suppose that α is in the image of the SLP. Define function $h : [0,1] \to G(\tilde{B})$ by $h(p) = G_{\boldsymbol{\theta}}(\mathbf{x}^p) = G(\boldsymbol{\theta}'\mathbf{x}^p)$ where $\mathbf{x}^p \in \tilde{B}$, i.e, $x_j^p = C_j^{-1}(\mathbb{1}_{\theta_j>0} \cdot p + \mathbb{1}_{\theta_j<0} \cdot (1 - p))$, $j = 1, 2, \ldots, k$. From the above argument, we have that $h(0) = G_{\boldsymbol{\theta}}(\mathbf{x}^0) \leq \alpha \leq G_{\boldsymbol{\theta}}(\mathbf{x}^1) = h(1)$. Since $G_{\boldsymbol{\theta}}(\cdot)$ and C^{-1} are continuous functions by **A1** and **A2**, h is also continuous. From the intermediate value theorem, there is a $p^* \in [0,1]$ such that $h(p^*) = \alpha$, which means that $G_{\boldsymbol{\theta}}(\mathbf{x}^{p^*}) = \alpha$. □

If we replace **A1** with the following more stringent assumption, the solution is unique.

A1'. All activation functions are strictly monotonic and continuous.

Proposition B.1.2 (Uniqueness of a Neutral Fair Baseline for SLPs)**.** *Given an SLP $G_{\boldsymbol{\theta}}(\cdot)$ satisfying **A1'**, a dataset satisfying **A2**, and a neutrality value α in the image of $G_{\boldsymbol{\theta}}(\cdot)$, then there exists a unique fair baseline x such that $G_{\boldsymbol{\theta}}(\mathbf{x}) = \alpha$.*

Proof. The existence has been shown in the previous Proposition. We prove

uniqueness by contradiction. Let us assume that there exists an $\mathbf{x}' \in \tilde{B}$ such that $G_{\boldsymbol{\theta}}(\mathbf{x}') = \alpha$ and $\mathbf{x}' \neq \mathbf{x}$. This means that there exists at least an element x'_i in the vector $\mathbf{x}'$ which is different from the element x_i of the vector $\mathbf{x}$. There are a total of four cases to analyse: $\theta_i > 0$ and $x'_i > x_i$, $\theta_i < 0$ and $x'_i > x_i$, $\theta_i > 0$ and $x'_i < x_i$, $\theta_i < 0$ and $x'_i < x_i$. We analyse only the case where $\theta_i > 0$ and $x'_i > x_i$, since the rest are fairly similar. $G_{\boldsymbol{\theta}}(\mathbf{x}') = G_{\boldsymbol{\theta}}([x'_1, \ldots, x'_i, ..., x'_k]) > G_{\boldsymbol{\theta}}([x'_1, ..., x_i, ..., x'_k])$, where the inequality follows from the strict monotonicity of $G_{\boldsymbol{\theta}}(\cdot)$, and that all the elements of the vector $\mathbf{x}'$ are kept constant except one. It is now worth observing that if $\theta_i > 0$ and $x'_i > x_i$, then $\mathbf{x}$ and $\mathbf{x}'$ to be in the set $\tilde{B}$ must be such that $x'_j \geq x_j \ \forall \theta_j > 0$ and $x'_j \leq x_j \ \forall \theta_j < 0$. Therefore, $G_{\boldsymbol{\theta}}(\mathbf{x}') > G_{\boldsymbol{\theta}}(\mathbf{x}) = \alpha$. □

B.2 Additional Analysis on the Synthetic Dataset

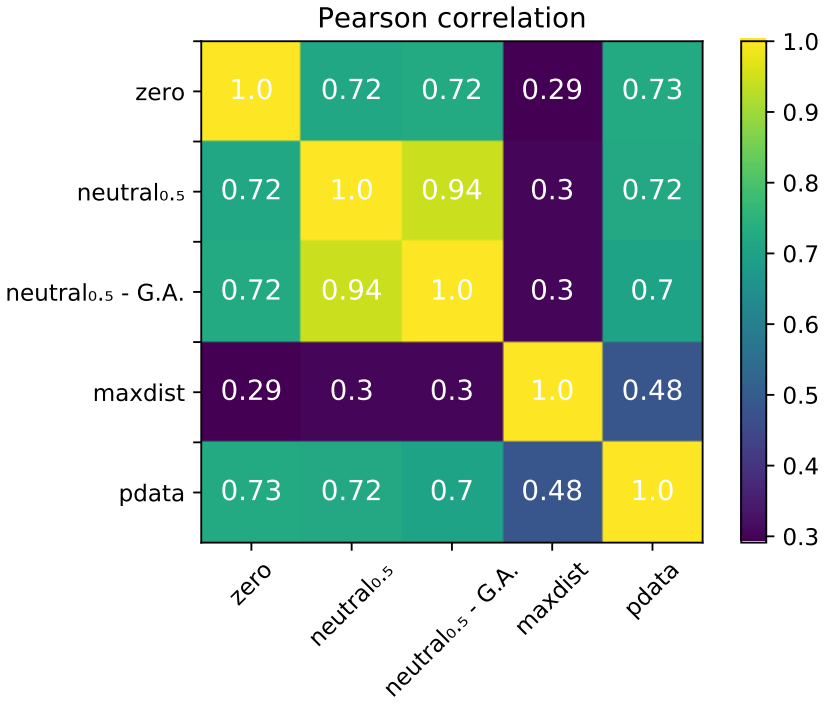


Figure B.1: Average Pearson correlation heatmap between attribution methods over the 100 draws of the Synthetic dataset.

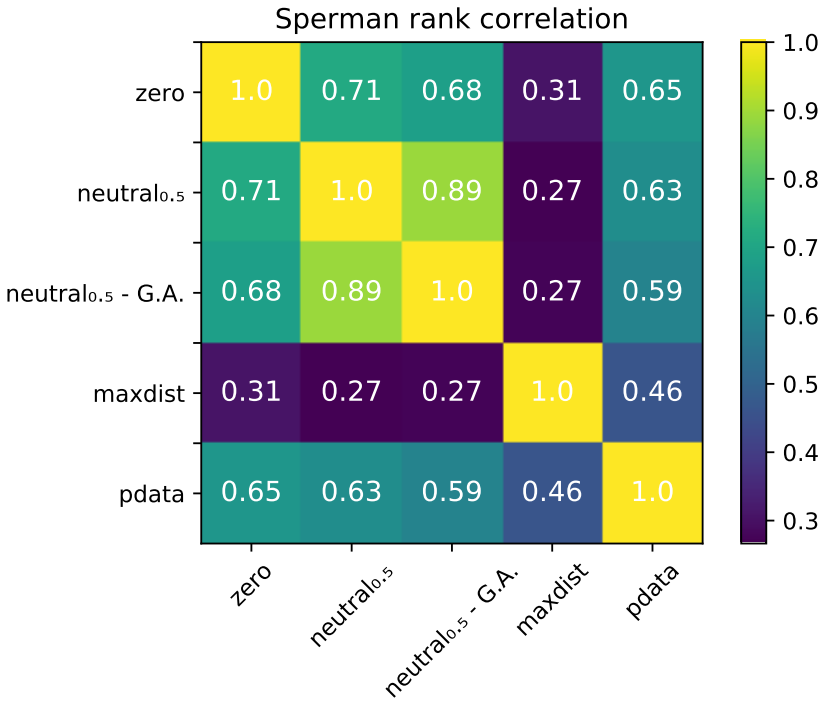


Figure B.2: Average Sperman correlation heatmap between attribution methods over the 100 draws of the Synthetic dataset.

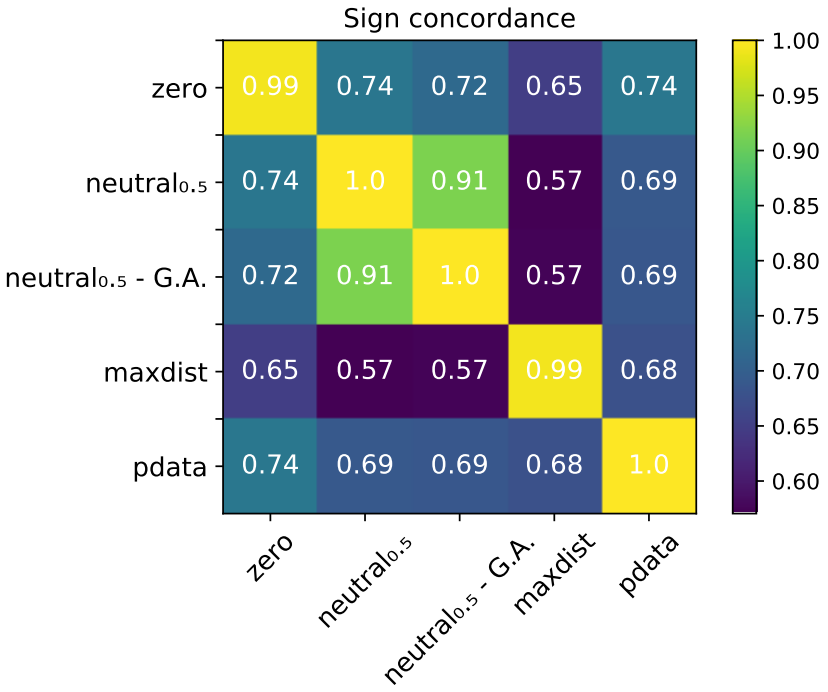


Figure B.3: Average sign concordance heatmap between attribution methods over the 100 draws of the Synthetic dataset.

Appendix C

Appendix to Chapter 6

C.1 Additional Experiment

In this section we compare different approaches on a dataset of weekly returns from the U.S. stock market. In particular, we restrict the investment set to the Standard & Poor's 500 sectoral indices, plus the Bloomberg US Treasury Index. The exercise is similar to the one presented in the main chapter, however we change the frequency from daily to weekly and the investment set by substituting the Bloomberg Barclay U.S. Aggregate Bond Index with the Bloomberg US Treasury Index. This exercise is aimed as a robustness check of the results presented in the main chapter, which we include in the table under the name daily to facilitate the comparison. A little of explanation is dutifully. In particular, the table shows that the performance metrics that are strongly consistent among the two experiments are turnover and maximum drawdown. The Sharpe ratio and Sortino ratio on the other side show some differences, this becomes particularly clear when looking at the Spearman correlation between the two out of sample evaluations. Nevertheless, the top strategy according to those metrics confirm to be the minimum variance, with the DLPO first second third fourth moment closely tracking, but this time at the third position instead of the second. Moreover, among the DLPO strategies this robustness check confirms that taking higher order moments into account when designing reward objectives may be beneficial.

Method	Sharpe Ratio		Sortino Ratio		Maximum Drawdown		Turnover	
	weekly	***daily***	***weekly***	***daily***	***weekly***	***daily***	***weekly***	***daily***
equally weighted	0.68	0.66	0.79	0.77	31.08	33.71	0.60	1.28
mean-variance	0.80	0.74	0.99	0.88	30.02	31.21	5.50	10.57
minimum variance	1.18	1.29	1.45	1.64	4.90	7.12	0.39	1.52
risk-parity	0.73	0.77	0.83	0.91	32.84	33.40	0.64	1.09
hierarchical risk-parity	0.95	0.89	1.26	1.10	30.81	32.09	0.29	0.34
DLPO sharpe	0.68	0.81	0.77	0.93	23.76	16.82	54.13	105.30
DLPO first moment	0.79	0.77	0.95	0.85	31.49	28.48	53.04	150.52
DLPO first second moment	0.64	0.79	0.76	0.90	27.05	28.35	53.69	142.24
DLPO first second third moment	0.68	1.04	0.83	1.41	26.99	21.47	45.75	114.00
DLPO first second third fourth moment	0.84	1.06	1.10	1.44	26.90	20.84	48.25	118.77
Pearson correlation	0.72		0.66		0.90		0.98	
Spearman correlation	0.36		0.50		0.90		0.82	

Table C.1: Recursive out-of-sample results of the different asset allocation strategies for the period 01/01/2011 to 15/03/2021. DLPO stands for Deep Learning Portfolio Optimisation methods. All the Deep Learning models are re-estimated yearly. Colours go from best blue to worst red. Pearson and Spearman correlations are computed between the daily and weekly performance metric.

Appendix D

Colophon

This document was set in the Times Roman typeface using LaTeX and BibTeX, composed with a text editor. The programming languages used for the empirical sections are Python 3.7.5 and Julia 1.0.5. Economic data are taken from Alfred in real time. Financial data are taken from Bloomberg and the UCI repository [280]. The source for data regarding the financial factors is AQR website: https://www.aqr.com/Insights/Datasets/Betting-Against-Beta-Equity-Factors-Daily.

Bibliography

[1] Saeed Nosratabadi, Amirhosein Mosavi, Puhong Duan, Pedram Ghamisi, Ferdinand Filip, Shahab S Band, Uwe Reuter, Joao Gama, and Amir H Gandomi. Data science in economics: comprehensive review of advanced machine learning and deep learning methods. *Mathematics*, 8(10):1799, 2020.

[2] Jian Huang, Junyi Chai, and Stella Cho. Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*, 14:1–24, 2020.

[3] Sebastian Doerr, Leonardo Gambacorta, José María Serena Garralda, et al. Big data and machine learning in central banking. *BIS Working Papers*, (930), 2021.

[4] John W Goodell, Satish Kumar, Weng Marc Lim, and Debidutta Pattnaik. Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. *Journal of Behavioral and Experimental Finance*, 32:100577, 2021.

[5] Halbert White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1(4):425–464, 1989.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[7] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

[8] Chung-Ming Kuan and Halbert White. Artificial neural networks: An econometric perspective. *Econometric reviews*, 13(1):1–91, 1994.

[9] Susan Athey. The impact of machine learning on economics. In *The economics of artificial intelligence*, pages 507–552. University of Chicago Press, 2019.

[10] Ragnar Frisch. Editor's note. *Econometrica*, 1(1):1–4, 1933.

[11] Susan Athey and Guido W Imbens. The state of applied econometrics: Causality and policy evaluation. *Journal of Economic Perspectives*, 31(2):3–32, 2017.

[12] Max H Farrell, Tengyuan Liang, and Sanjog Misra. Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213, 2021.

[13] Sendhil Mullainathan and Jann Spiess. Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106, 2017.

[14] Susan Athey and Guido W Imbens. Machine learning methods that economists should know about. *Annual Review of Economics*, 11:685–725, 2019.

[15] Marcos Lopez De Prado. Invited editorial comment: Mathematics and economics: A reality check, 2016.

[16] Daniele Bianchi, Matthias Büchner, and Andrea Tamoni. Bond risk premiums with machine learning. *The Review of Financial Studies*, 34(2):1046–1089, 2021.

[17] Thomas Cook and Aaron Smalter Hall. Macroeconomic indicator forecasting with deep neural networks. *Federal Reserve Bank of Kansas City, Research Working Paper*, (17-11), 2017.

[18] Julius Loermann and Benedikt Maas. Nowcasting us gdp with artificial neural networks. *Munich Personal RePEc Archive*, 2019.

[19] Markus Holopainen and Peter Sarlin. Toward robust early-warning models: A horse race, ensembles and model uncertainty. *Quantitative Finance*, 17(12):1933–1963, 2017.

[20] JB Heaton, Nicholas G. Polson, and Jan Hendrik Witte. Deep learning in finance. *arXiv preprint arXiv:1602.06561*, 2016.

[21] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *National Bureau of Economic Research*, 2018.

[22] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Autoencoder asset pricing models. *Journal of Econometrics*, 222(1):429–450, 2021.

[23] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep learning for portfolio optimization. *The Journal of Financial Data Science*, 2(4):8–20, 2020.

[24] Miquel Noguer i Alonso and Sonam Srivastava. Deep reinforcement learning for asset allocation in us equities. *arXiv preprint arXiv:2010.04404*, 2020.

[25] Giorgio Lucarelli and Matteo Borrotti. A deep q-learning portfolio management framework for the cryptocurrency market. *Neural Computing and Applications*, 32(23):17229–17244, 2020.

[26] Thomas E Koker and Dimitrios Koutmos. Cryptocurrency trading using machine learning. *Journal of Risk and Financial Management*, 13(8):178, 2020.

[27] Kumar Yashaswi. Deep reinforcement learning for portfolio optimization using latent feature state space (lfss) module. *arXiv preprint arXiv:2102.06233*, 2021.

[28] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[29] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2:2, 2017.

[30] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[31] Fama Eugene and Kenneth French. The cross-section of expected stock returns. *Journal of Finance*, 47(2):427–465, 1992.

[32] Eugene F Fama and R Kenneth. French, 1993, common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, 1993.

[33] Fama Eugene, R French Kenneth, et al. Multifactor explanations of asset pricing anomalies. *Journal of Finance*, 51(1):55–84, 1996.

[34] Mark M Carhart. On persistence in mutual fund performance. *The Journal of Finance*, 52(1):57–82, 1997.

[35] Andrea Frazzini and Lasse Heje Pedersen. Betting against beta. *Journal of Financial Economics*, 111(1):1–25, 2014.

[36] Catherine Doz, Domenico Giannone, and Lucrezia Reichlin. A two-step estimator for large approximate dynamic factor models based on kalman filtering. *Journal of Econometrics*, 164(1):188–205, 2011.

[37] Matteo Barigozzi and Matteo Luciani. Quasi maximum likelihood estimation and inference of large approximate dynamic factor models via the em algorithm. *arXiv preprint arXiv:1910.03821*, 2019.

[38] James H. Stock and Mark W. Watson. New indexes of coincident and leading economic indicators. *NBER macroeconomics annual*, 4:351–394, 1989.

[39] Marta Banbura and Michele Modugno. Maximum likelihood estimation of factor models on datasets with arbitrary pattern of missing data. *Journal of Applied Econometrics*, 29(1):133–160, 2014.

[40] Fernando GDC Ferreira, Amir H Gandomi, and Rodrigo TN Cardoso. Artificial intelligence applied to stock market trading: A review. *IEEE Access*, 9:30898–30917, 2021.

[41] Marcos Lopez De Prado. Building diversified portfolios that outperform out of sample. *The Journal of Portfolio Management*, 42(4):59–69, 2016.

[42] James B Heaton, Nick G Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

[43] Raúl Gómez Martínez, Miguel Prado Román, and Paola Plaza Casado. Big data algorithmic trading systems based on investors' mood. *Journal of Behavioral Finance*, 20(2):227–238, 2019.

[44] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[45] Patrick Houlihan and Germán G Creamer. Leveraging social media to predict continuation and reversal in asset prices. *Computational Economics*, 57(2):433–453, 2021.

[46] Tat Lung Chan and Nicholas Hale. Pricing european-type, early-exercise and discrete barrier options using an algorithm for the convolution of legendre series. *Quantitative Finance*, 20(8):1307–1324, 2020.

[47] Bo Gao. The use of machine learning combined with data mining technology in financial risk prevention. *Computational Economics*, pages 1–21, 2021.

[48] Qilun Li, Zhaoyi Xu, Xiaoqin Shen, and Jiacheng Zhong. Predicting business risks of commercial banks based on bp-ga optimized model. *Computational Economics*, pages 1–19, 2021.

[49] Adam Hale Shapiro, Moritz Sudhof, and Daniel J Wilson. Measuring news sentiment. *Journal of Econometrics*, 2020.

[50] Yan Chen, Zhilong Xie, Wenjie Zhang, Rong Xing, and Qing Li. Quantifying the effect of real estate news on chinese stock movements. *Emerging Markets Finance and Trade*, pages 1–26, 2020.

[51] Armando Marozzi. The ecb's tracker: Nowcasting the press conferences of the ecb. *ECB Working Paper*, 2021.

[52] Saule T Omarova. New tech v. new deal: Fintech as a systemic phenomenon. *Yale J. on Reg.*, 36:735, 2019.

[53] Julia Kokina, Ruth Gilleran, Shay Blanchette, and Donna Stoddard. Accountant as digital innovator: Roles and competencies in the age of automation. *Accounting Horizons*, 35(1):153–184, 2021.

[54] José Willer Prado, Valderí Castro Alcântara, Francisval Melo Carvalho, Kelly Carvalho Vieira, Luiz Kennedy Machado, and Dany Flávio Tonelli. Multivariate analysis of credit risk and bankruptcy research data: a bibliometric study involving different knowledge fields (1968—2014). *Scientometrics*, 106(3):1007–1029, 2016.

[55] Graham Elliott and Allan Timmermann. Forecasting in economics and finance. *Annual Review of Economics*, 8:81–110, 2016.

[56] Ashok K Nag and Amit Mitra. Forecasting daily foreign exchange rates using genetically optimized neural networks. *Journal of Forecasting*, 21(7):501–511, 2002.

[57] Javier Arroyo, Rosa Espínola, and Carlos Maté. Different approaches to forecast interval time series: a comparison in finance. *Computational Economics*, 37(2):169–191, 2011.

[58] Furao Shen, Jing Chao, and Jinxi Zhao. Forecasting exchange rate using deep belief networks and conjugate gradient method. *Neurocomputing*, 167:243–253, 2015.

[59] Vadlamani Ravi, Dadabada Pradeepkumar, and Kalyanmoy Deb. Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms. *Swarm and Evolutionary Computation*, 36:136–149, 2017.

[60] Svitlana Galeshchuk and Sumitra Mukherjee. Deep networks for predicting direction of change in foreign exchange rates. *Intelligent Systems in Accounting, Finance and Management*, 24(4):100–110, 2017.

[61] Lin Chen, Zhilin Qiao, Minggang Wang, Chao Wang, Ruijin Du, and Harry Eugene Stanley. Which artificial intelligence algorithm better predicts the chinese stock market? *IEEE Access*, 6:48625–48633, 2018.

[62] Ritika Singh and Shashi Srivastava. Stock prediction using deep learning. *Multimedia Tools and Applications*, 76(18):18569–18584, 2017.

[63] Jefferson Hernandez and Andres G Abad. Learning from multivariate discrete sequential data using a restricted boltzmann machine model. In *2018 IEEE 1st Colombian Conference on Applications in Computational Intelligence (ColCACI)*, pages 1–6. IEEE, 2018.

[64] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[65] Daigo Tashiro, Hiroyasu Matsushima, Kiyoshi Izumi, and Hiroki Sakaji. Encoding of high-frequency order information and prediction of short-term stock price by deep learning. *Quantitative Finance*, 19(9):1499–1506, 2019.

[66] Masaya Abe and Hideki Nakayama. Deep learning for forecasting stock returns in the cross-section. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 273–284. Springer, 2018.

[67] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

[68] Hongju Yan and Hongbing Ouyang. Financial time series prediction based on deep learning. *Wireless Personal Communications*, 102(2):683–700, 2018.

[69] Sotirios P Chatzis, Vassilis Siakoulis, Anastasios Petropoulos, Evangelos Stavroulakis, and Nikos Vlachogiannakis. Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert systems with applications*, 112:353–371, 2018.

[70] Mathias Kraus and Stefan Feuerriegel. Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, 104:38–48, 2017.

[71] Takashi Matsubara, Ryo Akita, and Kuniaki Uehara. Stock price prediction by deep neural generative model of news articles. *IEICE TRANSACTIONS on Information and Systems*, 101(4):901–908, 2018.

[72] Dang Lien Minh, Abolghasem Sadeghi-Niaraki, Huynh Duc Huy, Kyungbok Min, and Hyeonjoon Moon. Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. *IEEE Access*, 6:55392–55404, 2018.

[73] Ying L Becker and Marc R Reinganum. *The current state of quantitative equity investing.* 2018.

[74] Pei-Ying Hsu, Chin Chou, Szu-Hao Huang, and An-Pin Chen. A market making quotation strategy based on dual deep learning agents for option pricing and bid-ask spread estimation. In *2018 IEEE international conference on agents (ICA)*, pages 99–104. IEEE, 2018.

[75] Nhi NY Vo, Xuezhong He, Shaowu Liu, and Guandong Xu. Deep learning for decision making and the optimization of socially responsible investments and portfolio. *Decision Support Systems*, 124:113097, 2019.

[76] Frank Smets, Kai Christoffel, Günter Coenen, Roberto Motto, and Massimo Rostagno. Dsge models and their use at the ecb. *SERIEs*, 1(1):51–65, 2010.

[77] Frank Smets and Rafael Wouters. Shocks and frictions in us business cycles: A bayesian dsge approach. *American Economic Review*, 97(3):586–606, 2007.

[78] Marco Del Negro, Stefano Eusepi, Marc P Giannoni, Argia M Sbordone, Andrea Tambalotti, Matthew Cocci, Raiden Hasegawa, and M Linder. The frbny dsge model. *FRB of New York Staff Report*, (647), 2013.

[79] Lawrence J Christiano, Roberto Motto, and Massimo Rostagno. Risk shocks. *American Economic Review*, 104(1):27–65, 2014.

[80] Günter Coenen, Peter Karadi, Sebastian Schmidt, and Anders Warne. The new area-wide model ii: an extended version of the ecb's micro-

founded model for forecasting and policy analysis with a financial sector. *ECB working paper*, 2018.

[81] Raffaella Giacomini. Economic theory and forecasting: lessons from the literature. *The Econometrics Journal*, 18(2):C22–C41, 06 2015.

[82] Jesús Fernández-Villaverde and Pablo A Guerrón-Quintana. Estimating dsge models: Recent advances and future challenges. *Annual Review of Economics*, 13:229–252, 2021.

[83] John Geweke. The dynamic factor analysis of economic time series. *Latent Variables in Socio-Economic Models*, 1977.

[84] Thomas Sargent and C.A. Sims. Business cycle modeling without pretending to have too much a priori economic theory. *New Methods in Business Cycle Research*, pages 45–109, 10 1977.

[85] James H. Stock and Mark W. Watson. Forecasting using principal components from a large number of predictors. *Journal of American Statistical Association*, 97:1167–1179, 2002.

[86] James H. Stock and Mark W. Watson. Macroeconomic forecasting using diffusion indexes. *Journal of Business Economics and Statistics*, 20:147–162, 2002.

[87] Mario Forni and Marco Lippi. The generalized dynamic factor model: representation theory. *Econometric theory*, 17(6):1113–1141, 2001.

[88] Mario Forni, Marc Hallin, Marco Lippi, and Lucrezia Reichlin. The generalized dynamic-factor model: Identification and estimation. *Review of Economics and Statistics*, 82(4):540–554, 2000.

[89] Mario Forni, Marc Hallin, Marco Lippi, and Lucrezia Reichlin. The generalized dynamic factor model: one-sided estimation and forecasting. *Journal of the American Statistical Association*, 100(471):830–840, 2005.

[90] Mario Forni, Marc Hallin, Marco Lippi, and Paolo Zaffaroni. Dynamic factor models with infinite-dimensional factor spaces: One-sided representations. *Journal of Econometrics*, 185(2):359–371, 2015.

[91] Mario Forni, Alessandro Giovannelli, Marco Lippi, and Stefano Soccorsi. Dynamic factor model with infinite-dimensional factor space: Forecasting. *Journal of Applied Econometrics*, 33(5):625–642, 2018.

[92] Filippo Altissimo, Riccardo Cristadoro, Mario Forni, Marco Lippi, and Giovanni Veronese. New eurocoin: Tracking economic growth in real time. *Review of Economics and Statistics*, 92(4):1024–1034, 2010.

[93] Marta Banbura, Domenico Giannone, and Lucrezia Reichlin. Nowcasting. *ECB working paper*, 2010.

[94] Gary Koop and Dimitris Korobilis. *Bayesian multivariate time series methods for empirical macroeconomics*. Now Publishers Inc, 2010.

[95] Andrea Carriero, George Kapetanios, and Massimiliano Marcellino. Forecasting large datasets with bayesian reduced rank multivariate models. *Journal of Applied Econometrics*, 26(5):735–761, 2011.

[96] Andrea Carriero, George Kapetanios, and Massimiliano Marcellino. Structural analysis with classical and bayesian large reduced rank vars. *Mimeo*, 2014.

[97] Jushan Bai and Peng Wang. Identification and bayesian estimation of dynamic factor models. *Journal of Business & Economic Statistics*, 33(2):221–240, 2015.

[98] Antonello D'Agostino, Domenico Giannone, Michele Lenza, and Michele Modugno. Nowcasting business cycles: A bayesian approach to dynamic heterogeneous factor models. In *Dynamic Factor Models*. Emerald Group Publishing Limited, 2016.

[99] Siem Jan Koopman and Geert Mesters. Empirical bayes methods for dynamic factor models. *Review of Economics and Statistics*, 99(3):486–498, 2017.

[100] Juan Antolin-Diaz, Thomas Drechsel, and Ivan Petrella. Advances in nowcasting economic activity: Secular trends, large shocks and new data. *CEPR Discussion Paper No. DP15926*, 2021.

[101] Catherine Doz and Peter Fuleky. Dynamic factor models. *Macroeconomic forecasting in the era of big data*, pages 27–64, 2020.

[102] Mr Maxym Kryshko. *Data-rich DSGE and dynamic factor models*. International Monetary Fund, 2011.

[103] Jean Boivin and Marc P Giannoni. Has monetary policy become more effective? *Review of Economics and Statistics*, 88(3):445–462, 2006.

[104] James H Stock and Mark Watson. Forecasting in dynamic factor models subject to structural instability. *The Methodology and Practice of Econometrics. A Festschrift in Honour of David F. Hendry*, 173:205, 2009.

[105] Philippe Goulet Coulombe. A neural phillips curve and a deep output gap. *arXiv preprint arXiv:2202.04146*, 2022.

[106] Niko Hauzenberger, Florian Huber, and Karin Klieber. Real-time inflation forecasting using non-linear dimension reduction techniques. *arXiv preprint arXiv:2012.08155*, 2020.

[107] Radu Gabriel Cristea. Can alternative data improve the accuracy of dynamic factor model nowcasts? *Faculty of Economics, University of Cambridge*, 2020.

[108] Merton H Miller. The history of finance. *The Journal of Portfolio Management*, 25(4):95–101, 1999.

[109] Campbell R Harvey and Yan Liu. A census of the factor zoo. *Available at SSRN 3341728*, 2019.

[110] Kerry Back. *Asset pricing and portfolio choice theory*. Oxford University Press, 2017.

[111] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[112] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[113] Mohit Sewak. *Deep Reinforcement Learning*. Springer, 2019.

[114] Yuh-Jong Hu and Shang-Jen Lin. Deep reinforcement learning for optimizing finance portfolio management. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 14–20. IEEE, 2019.

[115] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83, 2018.

[116] David H Bailey, Jonathan Borwein, Marcos Lopez de Prado, and Qiji Jim Zhu. The probability of backtest overfitting. *Journal of Computational Finance, forthcoming*, 2016.

[117] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[118] Saud Almahdi and Steve Y Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, 2017.

[119] Dimitri P Bertsekas and Steven E Shreve. *Stochastic optimal control: the discrete-time case*, volume 5. Athena Scientific, 1996.

[120] John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(5-6):441–470, 1998.

[121] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4):875–889, 2001.

[122] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*, 2018.

[123] Chiao-Ting Chen, An-Pin Chen, and Szu-Hao Huang. Cloning strategies from trading records using agent-based reinforcement learning algorithm. In *2018 IEEE International Conference on Agents (ICA)*, pages 34–37. IEEE, 2018.

[124] Souradeep Chakraborty. Capturing financial markets to apply deep reinforcement learning. *arXiv preprint arXiv:1907.04373*, 2019.

[125] Gyeeun Jeong and Ha Young Kim. Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, 117:125–138, 2019.

[126] Gordon Ritter. Machine learning for trading. *Available at SSRN 3015609*, 2017.

[127] Qiang Song, Anqi Liu, and Steve Y Yang. Stock portfolio selection using learning-to-rank algorithms with news sentiment. *Neurocomputing*, 264:20–28, 2017.

[128] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *ArXiv*, abs/1706.10059, 2017.

[129] Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*, 2018.

[130] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, 2016.

[131] Wuyu Wang, Weizi Li, Ning Zhang, and Kecheng Liu. Portfolio formation with preselection using deep learning from long-term financial data. *Expert Systems with Applications*, 143:113042, 2020.

[132] Thomas Raffinot. Hierarchical clustering-based asset allocation. *The Journal of Portfolio Management*, 44(2):89–99, 2017.

[133] Rafael Moral-Escudero, Rubén Ruiz-Torrubiano, and Alberto Suárez. Selection of optimal investment portfolios with cardinality constraints. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2382–2388. IEEE, 2006.

[134] Ghada Hassan and Christopher D Clack. Multiobjective robustness for portfolio optimization in volatile environments. In *Proceedings of the 10th annual conference on Genetic and Evolutionary Computation*, pages 1507–1514, 2008.

[135] Gili Rosenberg, Poya Haghnegahdar, Phil Goddard, Peter Carr, Kesheng Wu, and Marcos López De Prado. Solving the optimal trading trajectory problem using a quantum annealer. *IEEE Journal of Selected Topics in Signal Processing*, 10(6):1053–1060, 2016.

[136] Kei Nakagawa, Takumi Uchida, and Tomohisa Aoshima. Deep factor model. In *ECML PKDD 2018 Workshops*, pages 37–50. Springer, 2018.

[137] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015.

[138] Kei Nakagawa, Tomoki Ito, Masaya Abe, and Kiyoshi Izumi. Deep recurrent factor model: interpretable non-linear and time-varying multi-factor model. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2019.

[139] Philippe Bracke, Anupam Datta, Carsten Jung, and Shayak Sen. Machine learning explainability in finance: an application to default risk analysis. *Bank of England Working Paper*, 2019.

[140] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE Symposium on Security and Privacy*, pages 598–617. IEEE, 2016.

[141] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. Explainable machine learning in credit risk management. *Computational Economics*, 57(1):203–216, 2021.

[142] Andreas GF Hoepner, David McMillan, Andrew Vivian, and Chardin Wese Simen. Significance, relevance and explainability in the machine learning age: an econometrics and financial data science perspective. *The European Journal of Finance*, 27(1-2):1–7, 2021.

[143] C. Spearman. General intelligence objectively determined and measured. *American Journal of Psychology*, 15:201–293, 1904.

[144] Arthur F. Burns and Wesley C. Mitchell. *Measuring Business Cycles*. Number burn46-1 in NBER Books. National Bureau of Economic Research, Inc, July 1946.

[145] James H Stock and Mark W Watson. A probability model of the coincident economic indicators. *NBER Working Paper*, 1988.

[146] James H Stock, Mark W Watson, et al. A procedure for predicting recessions with leading indicators: Econometric issues and recent experience. *Business cycles, indicators and forecasting*, 28:95–156, 1993.

[147] Gary Chamberlain. Funds, factors, and diversification in arbitrage pricing models. *Econometrica*, pages 1305–1323, 1983.

[148] Gary Chamberlain and Michael Rothschild. Arbitrage, factor structure, and mean-variance analysis on large asset markets. *Econometrica*, pages 1281–1304, 1983.

[149] James H. Stock and Mark W. Watson. Dynamic factor models. *Oxford Handbooks Online*, 2011.

[150] James H Stock and Mark W Watson. Dynamic factor models, factor-augmented vector autoregressions, and structural vector autoregressions in macroeconomics. In *Handbook of Macroeconomics*, volume 2, pages 415–525. Elsevier, 2016.

[151] Matteo Barigozzi. Dynamic factor models. *Lecture notes. London School of Economics*, 2018.

[152] Jushan Bai and Serena Ng. Determining the number of factors in approximate factor models. *Econometrica*, 70(1):191–221, 2002.

[153] Mark W. Watson and Robert F. Engle. Alternative algorithms for the estimation of dynamic factor, mimic and varying coefficient regression models. *Journal of Econometrics*, 23(3):385–400, 1983.

[154] Catherine Doz, Domenico Giannone, and Lucrezia Reichlin. A quasi–maximum likelihood approach for large, approximate dynamic factor models. *Review of Economics and Statistics*, 94(4):1014–1024, 2012.

[155] Jörg Breitung and Jörn Tenhofen. Gls estimation of dynamic factor models. *Journal of the American Statistical Association*, 106(495):1150–1166, 2011.

[156] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[157] Tommaso Proietti. Estimation of common factors under cross-sectional and temporal aggregation constraints: nowcasting monthly gdp and its main components. In *COMPSTAT 2008*, pages 547–558. Springer, 2008.

[158] Borus Jungbacker and Siem Jan Koopman. Likelihood-based analysis for dynamic factor models. Technical report, Tinbergen Institute Discussion Paper, 2008.

[159] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[160] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[161] Yann LeCun. Phd thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models). 1987.

[162] Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[163] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[164] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[165] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[166] Tijmen Tieleman and Geoffrey Hinton. Rmsprop. *University of Toronto, Technical Report*, 7017, 2012.

[167] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[168] Robin M Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley-benchmarking deep learning optimizers. *International Conference on Machine Learning*, pages 9367–9376, 2021.

[169] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19:143–155, 1989.

[170] Yann Le Cun, Conrad C Galland, and Geoffrey E Hinton. Gemini: gradient estimation through matrix inversion after noise injection. In *Advances in Neural Information Processing Systems*, pages 141–148, 1989.

[171] Yann Le Cun, Lionel D Jackel, Brian Boser, John S Denker, Henry P Graf, Isabelle Guyon, Don Henderson, Richard E Howard, and William Hubbard. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

[172] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[173] Yann Le Cun, Ofer Matan, Bernhard Boser, John S Denker, Don Henderson, Richard E Howard, Wayne Hubbard, LD Jacket, and Henry S Baird. Handwritten zip code recognition with multilayer networks. *International Conference on Pattern Recognition*, 2:35–40, 1990.

[174] Patrice Simard and Yann Le Cun. Reverse tdnn: an architecture for trajectory generation. In *Advances in Neural Information Processing Systems*, pages 579–588, 1992.

[175] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. *International Conference on Machine Learning*, pages 399–406, 2010.

[176] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1279–1287, 2010.

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

[178] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

[179] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[180] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[181] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. *International Conference on Machine Learning*, pages 2067–2075, 2015.

[182] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. *International conference on Machine Learning*, pages 2342–2350, 2015.

[183] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

[184] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Advances in Neural Information Processing Systems*, pages 3–10, 1994.

[185] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[186] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[187] Nathalie Japkowicz, Stephen Jose Hanson, and Mark A Gluck. Nonlinear autoassociation is not equivalent to pca. *Neural Computation*, 12(3):531–545, 2000.

[188] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[189] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[190] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551 – 560, 1990.

[191] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pages 6231–6239, 2017.

[192] Angshul Majumdar and Aditay Tripathi. Asymmetric stacked autoencoder. pages 911–918. IEEE, 2017 International Joint Conference on Neural Networks (IJCNN), 2017.

[193] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine learning*, pages 1096–1103, 2008.

[194] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907, 2013.

[195] Yichuan Tang and Chris Eliasmith. Deep networks for robust visual recognition. *International Conference on Machine Learning*, pages 1055–1062, 2010.

[196] Navdeep Jaitly and Geoffrey E. Hinton. Vocal tract length perturbation (vtlp) improves speech recognition. volume 117. Proc. ICML Workshop on Deep Learning for Audio, Speech and Language, 2013.

[197] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

[198] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2(5):6, 2017.

[199] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[200] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1242–1250. PMLR, 2014.

[201] Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder. *International Conference on Learning Representations*, 2019.

[202] Rahul G. Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2101–2109. AAAI Press, 2017.

[203] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3601–3610, 2017.

[204] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

[205] Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2946–2954. Curran Associates, Inc., 2016.

[206] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. *International Conference on Machine Learning*, pages 387–395, 2014.

[207] Bernd Scherer and Kenneth Winston. *The Oxford handbook of quantitative asset management*. OUP Oxford, 2011.

[208] Otto Loistl. The erroneous approximation of expected utility by means of a taylor's series expansion: analytic and computational results. *The American Economic Review*, 66(5):904–910, 1976.

[209] Francois-Serge Lhabitant. On the (ab) use of taylor series approximations for portfolio selection, portfolio performance and risk management. *HEC, University of Lausanne*, 1998.

[210] Eric Jondeau and Michael Rockinger. Optimal portfolio allocation under higher moments. *European Financial Management*, 12(1):29–55, 2006.

[211] Robert C Scott and Philip A Horvath. On the direction of preference for moments of higher order than the variance. *The Journal of Finance*, 35(4):915–919, 1980.

[212] Gary Chamberlain. A characterization of the distributions that imply mean—variance utility functions. *Journal of Economic Theory*, 29(1):185–201, 1983.

[213] Yoram Kroll, Haim Levy, and Harry M Markowitz. Mean-variance versus direct utility maximization. *The Journal of Finance*, 39(1):47–61, 1984.

[214] Campbell R Harvey, John C Liechty, Merrill W Liechty, and Peter Müller. Portfolio selection with higher moments. *Quantitative Finance*, 10(5):469–485, 2010.

[215] Frank J Fabozzi, Petter N Kolm, Dessislava A Pachamanova, and Sergio M Focardi. *Robust portfolio optimization and management*. John Wiley & Sons, 2007.

[216] Kin Keung Lai, Lean Yu, and Shouyang Wang. Mean-variance-skewness-kurtosis-based portfolio optimization. In *First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)*, volume 2, pages 292–297. IEEE, 2006.

[217] Walter Sun, Ayres Fan, Li-Wei Chen, Tom Schouwenaars, and Marius A Albota. Optimal rebalancing for institutional portfolios. *The Journal of Portfolio Management*, 32(2):33–43, 2006.

[218] Malcolm Baker, Brendan Bradley, and Jeffrey Wurgler. Benchmarks as limits to arbitrage: Understanding the low-volatility anomaly. *Financial Analysts Journal*, 67(1):40–54, 2011.

[219] Xi Bai, Katya Scheinberg, and Reha Tutuncu. Least-squares approach to risk parity in portfolio selection. *Quantitative Finance*, 16(3):357–376, 2016.

[220] Marcos Lopez De Prado. *Machine learning for asset managers*. Cambridge University Press, 2020.

[221] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[222] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019.

[223] Gregor Stiglic, Primoz Kocbek, Nino Fijacko, Marinka Zitnik, Katrien Verbert, and Leona Cilar. Interpretability of machine learning based prediction models in healthcare. *arXiv preprint arXiv:2002.08596*, 2020.

[224] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[225] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

[226] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013.

[227] Neil Gallagher, Kyle R Ulrich, Austin Talbot, Kafui Dzirasa, Lawrence Carin, and David E Carlson. Cross-spectral factor analysis. In *Advances in Neural Information Processing Systems*, pages 6842–6852, 2017.

[228] Chaofan Chen, Kangcheng Lin, Cynthia Rudin, Yaron Shaposhnik, Sijia Wang, and Tong Wang. An interpretable model with globally consistent explanations for credit risk. *arXiv preprint arXiv:1811.12615*, 2018.

[229] Fulton Wang, Cynthia Rudin, Tyler H Mccormick, and John L Gore. Modeling recovery curves with application to prostatectomy. *Biostatistics*, 20(4):549–564, 2019.

[230] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *International Conference on Machine Learning*, pages 3319–3328, 2017.

[231] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[232] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[233] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

[234] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

[235] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[236] M. Ancona, C. Öztireli, and M. Gross. Explaining deep neural networks with a polynomial time algorithm for shapley values approximation. *International Conference on Machine Learning*, 2019.

[237] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

[238] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[239] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *International Conference on Machine Learning-Volume 70*, pages 3145–3153, 2017.

[240] Matthew Zeiler and Rob Fergus. Visualizing and understanding convo-

lutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[241] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[242] Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*, 2019.

[243] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. Xrai: Better attributions through regions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4948–4957, 2019.

[244] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *arXiv preprint arXiv:1908.08474*, 2019.

[245] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

[246] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[247] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[248] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9734–9745, 2019.

[249] M Ancona, E Ceolini, C Öztireli, and M Gross. A unified view of gradient-based attribution methods for deep neural networks. *NIPS 2017-Workshop on Interpreting, Explaining and Visualizing Deep Learning*, 2017.

[250] Michael W. McCracken and Serena Ng. Fred-md: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4):574–589, 2016.

[251] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[252] Roberto S. Mariano and Yasutomo Murasawa. A new coincident index of business cycles based on monthly and quarterly series. *Journal of Applied Econometrics*, 18(4):427–443, 2003.

[253] Robert Engle and Mark W. Watson. A one-factor multivariate time series model of metropolitan wage rates. *Journal of the American Statistical Association*, 76(376):774–781, 1981.

[254] Mario Forni and Lucrezia Reichlin. Let's get real: a factor analytical approach to disaggregated business cycle dynamics. *The Review of Economic Studies*, 65(3):453–473, 1998.

[255] Domenico Giannone, Lucrezia Reichlin, and David Small. Nowcasting: The real-time informational content of macroeconomic data. *Journal of Monetary Economics*, 55(4):665–676, 2008.

[256] Borus Jungbacker, Siem Jan Koopman, and Michel Van der Wel. Maxi-

mum likelihood estimation for dynamic factor models with missing data. *Journal of Economic Dynamics and Control*, 35(8):1358–1368, 2011.

[257] Anders Bredahl Kock, Timo Teräsvirta, et al. Forecasting with nonlinear time series models. *Oxford handbook of economic forecasting*, pages 61–87, 2011.

[258] William A. Barnett, Marcelle Chauvet, and Danilo Leiva-Leon. Real-time nowcasting of nominal gdp with structural breaks. *Journal of Econometrics*, 191(2):312–324, 2016.

[259] Maximo Camacho, Gabriel Perez-Quiros, and Pilar Poncela. Markov-switching dynamic factor models in real time. 2012.

[260] Massimiliano Marcellino and Christian Schumacher. Factor midas for nowcasting and forecasting with ragged-edge data: A model comparison for german gdp. *Oxford Bulletin of Economics and Statistics*, 72(4):518–550, 2010.

[261] Dimitris Korobilis. Forecast comparison of nonlinear time series models of us gdp: A bayesian approach. *Available at SSRN 1508486*, 2006.

[262] Jouchi Nakajima and Mike West. Bayesian analysis of latent threshold dynamic models. *Journal of Business & Economic Statistics*, 31(2):151–164, 2013.

[263] Gianni Amisano and Oreste Tristani. Exact likelihood computation for nonlinear dsge models with heteroskedastic innovations. *Journal of Economic Dynamics and Control*, 35(12):2167–2185, 2011.

[264] Jushan Bai and Serena Ng. Forecasting economic time series using targeted predictors. *Journal of Econometrics*, 146(2):304 – 317, 2008. Honoring the research contributions of Charles R. Nelson.

[265] Eli Stevens and Luca Antiga. *Deep Learning with PyTorch*. Manning Publications, 2019.

[266] James H. Stock and Mark W. Watson. Chapter 8 - dynamic factor models, factor-augmented vector autoregressions, and structural vector autoregressions in macroeconomics. volume 2 of *Handbook of Macroeconomics*, pages 415 – 525. Elsevier, 2016.

[267] M. Hashem Pesaran and Simon M. Potter. Nonlinear dynamics and econometrics: An introduction. *Journal of Applied Econometrics*, 7:S1–S7, 1992.

[268] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[269] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? pages 2483–2493. Advances in Neural Information Processing Systems, 2018.

[270] Lucia Alessi, Matteo Barigozzi, and Marco Capasso. Improved penalization for determining the number of factors in approximate factor models. *Statistics & Probability Letters*, 80(23-24):1806–1813, 2010.

[271] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. pages 2546–2554, 2011.

[272] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[273] Jennie Bai, Eric Ghysels, and Jonathan H Wright. State space models and midas regressions. *Econometric Reviews*, 32(7):779–813, 2013.

[274] Claudia Foroni and Massimiliano Giuseppe Marcellino. A survey of econometric methods for mixed-frequency data. *Available at SSRN 2268912*, 2013.

[275] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[276] Jan Hannes Lang, Cosimo Izzo, Stephan Fahr, and Josef Ruzicka. Anticipating the bust: a new cyclical systemic risk indicator to assess the likelihood and severity of financial crises. *ECB Occasional Paper*, (219), 2019.

[277] Roger Koenker and Gilbert Jr Bassett. Regression quantiles. *Econometrica*, pages 33–50, 1978.

[278] Liang Chen, Juan J Dolado, and Jesús Gonzalo. Quantile factor models. *Econometrica*, 89(2):875–910, 2021.

[279] Lucia Alessi and Carsten Detken. Quasi real time early warning indicators for costly asset price boom/bust cycles: A role for global liquidity. *European Journal of Political Economy*, 27(3):520–533, 2011.

[280] I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480, 2009.

[281] James J Angel, Todd J Broms, and Gary L Gastineau. Etf transaction costs are often higher than investors realize. *The Journal of Portfolio Management*, 42(3):65–75, 2016.

[282] Nicholas Moehle, Stephen Boyd, and Andrew Ang. Portfolio performance attribution via shapley value. *arXiv preprint arXiv:2102.05799*, 2021.

[283] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. *International Conference on Learning Representations*, 2019.

[284] Emanuele Albini, Jason Long, Danial Dervovic, and Daniele Magazzeni. Counterfactual shapley additive explanations. *ACM Conference on Fairness, Accountability, and Transparency*, pages 1054–1070, 2022.

[285] Tobias Adrian, Nina Boyarchenko, and Domenico Giannone. Vulnerable growth. *American Economic Review*, 109(4):1263–89, 2019.

[286] Lucrezia Reichlin, Giovanni Ricco, and Thomas Hasenzagl. Financial variables as predictors of real growth vulnerability. *Discussion Paper*, 2020.

[287] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *International Conference on Learning Representations*, 2022.

[288] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *International Conference on Machine Learning*, pages 12310–12320, 2021.

[289] Filippo Pellegrino. Selecting time-series hyperparameters with the artificial jackknife. *arXiv preprint arXiv:2002.04697*, 2020.

[290] Simon Haykin. *Kalman filtering and neural networks*, volume 47. John Wiley & Sons, 2004.

[291] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*. Oxford university press, 2012.

[292] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 82:34–45, 1960.

[293] AH Jazwinski. Stochastic process and filtering theory. *New York: Academic Press*, 1970.

[294] Peter S Maybeck. *Stochastic models, estimation, and control*. New York: Academic Press, 1982.

[295] Simon J Julier, Jeffrey K Uhlmann, and Hugh F Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of 1995 American Control Conference-ACC'95*, volume 3, pages 1628–1632. IEEE, 1995.

[296] Simon Julier and Jeffrey K Uhlmann. A general method for approximating nonlinear transformations of probability distributions. 1996.

[297] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics, 1997.

[298] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[299] Rudolph Van Der Merwe and Eric A Wan. Efficient derivative-free kalman filters for online learning. In *Proceedings of European Symposium on Artificial Neural Networks*, pages 205–210. ESANN, 2001.

[300] Kazufumi Ito and Kaiqi Xiong. Gaussian filters for nonlinear filtering problems. *IEEE transactions on automatic control*, 45(5):910–927, 2000.

[301] Ashkan Khakzar, Soroosh Baselizadeh, Saurabh Khanduja, Christian Rupprecht, Seong Tae Kim, and Nassir Navab. Neural response interpretation through the lens of critical pathways. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13528–13538, 2021.