Insper
Programa de Mestrado Profissional em Economia

Alexandre Fernandes Theoharidis

Forecasting Inflation Using Deep Learning: An Application of Convolutional LSTM Networks and Variational Autoencoders

São Paulo
2021

Alexandre Fernandes Theoharidis

Forecasting Inflation Using Deep Learning: An Application of Convolutional LSTM Networks and Variational Autoencoders

Dissertação apresentada ao Programa de Mestrado Profissional em Economia do Insper como parte dos requisitos para obtenção do título de Mestre em Economia.

Área de concentração: Economia dos Negócios

Linha de pesquisa: Macroeconomia

Insper
Programa de Mestrado Profissional em Economia

Supervisor: Diogo Abry Guillén
Co-supervisor: Hedibert Freitas Lopes

São Paulo
2021

Theoharidis, Alexandre Fernandes.

Forecasting Inflation Using Deep Learning: An Application of Convolutional LSTM Networks and Variational Autoencoders. / Alexandre Fernandes Theoharidis. – São Paulo, 2021.

153f.

Dissertação (Mestrado) – Insper
Programa de Mestrado Profissional em Economia, 2021.

Orientador: Diogo Abry Guillén
Coorientador: Hedibert Freitas Lopes

1. Deep Learning. 2. Machine Learning. 3. Inflation Forecasting. 4. LSTM Networks. 5. Convolutional Networks. 6. Autoencoders. I. Diogo Abry Guillén. II. Hedibert Freitas Lopes. III. Insper. IV. Departamento de Economia. V. Forecasting Inflation Using Deep Learning: An Application of Convolutional LSTM Networks and Variational Autoencoders.

Alexandre Fernandes Theoharidis

Forecasting Inflation Using Deep Learning: An Application of Convolutional LSTM Networks and Variational Autoencoders

Dissertação apresentada ao Programa de Mestrado Profissional em Economia do Insper como parte dos requisitos para obtenção do título de Mestre em Economia.

Área de concentração: Economia dos Negócios

Linha de pesquisa: Macroeconomia

DATA DE APROVAÇÃO: São Paulo, 08 de fevereiro de 2021.

BANCA EXAMINADORA

Diogo Abry Guillén
Orientador

Hedibert Freitas Lopes
Coorientador

Miguel Maria Charters de Oliveira Bandeira da Silva
Convidado Interno

Flavio Almeida de Magalhães Cipparrone
Convidado Externo

À minha família e amigos.

AGRADECIMENTOS

Primeiramente, agradeço ao meu orientador, professor Diogo Abry Guillén, e ao meu coorientador, professor Hedibert Freitas Lopes, por me acompanharem e me instruírem ao decorrer da elaboração dessa dissertação. Foi uma honra. Nossas interações foram fundamentais para o sucesso desse trabalho, e as sugestões e conselhos recebidos contribuíram substancialmente para a sua qualidade. Ademais, ambos me estimularam a seguir em frente sempre que preciso e a atingir objetivos arrojados e desafiadores. São professores brilhantes e exemplares. Seus ensinamentos foram incomensuráveis.

Concomitantemente, quero agradecer aos amigos que formei ao decorrer do mestrado e que carregarei comigo para o restante da minha vida. Foram um verdadeiro presente. Nossa convivência ao longo dessa jornada foi gratificante e prazerosa. Sem dúvida alguma, tornou o mestrado não apenas mais leve, mas também enriqueceu minha experiência e aprendizagem. Nossos debates, estudos e trabalhos engrandeceram o curso e me motivaram continuamente. Essa dissertação também tem a contribuição de todos vocês.

Em particular, preciso agradecer à Nelly Colnaghi, uma amiga incrível e muito especial. Sua parceria nos projetos e estudos foi espetacular, sempre me estimulando, me ensinando, me animando e me ajudando, mesmo nos momentos mais difíceis, e não foram poucos. Ainda em especial, deixo aqui meu agradecimento a Gabriel, Débora e Milena, três grandes amigos que estiveram ao meu lado sempre que necessitei e com os quais estudos, discussões e projetos agregaram enormemente para mim. Nossos momentos de descontração, convivência e apoio foram sensacionais. Não menos importantes, preciso citar nominalmente também Rafael Couto, Ivy, Eduardo, Fernando Casagrande e Lucas Avelleda. Foi um prazer cursar esse mestrado com todos vocês. Obrigado por tudo. Vocês poderão contar comigo sempre.

Ademais, gostaria de agradecer ao Insper, que apresentou um corpo docente de excelência com o qual tive a oportunidade de estudar ao longo dessa trajetória. São profissionais comprometidos, diligentes e dedicados, que transmitiram conhecimentos valiosos. Também destaco a qualidade da infraestrutura da universidade, que ofereceu todos os recursos necessários para prover uma educação de alto nível, mesmo durante a pandemia do Covid-19, e o suporte da coordenadoria, que esteve sempre presente.

Finalmente, agradeço à minha família pela educação, instrução e valores que me proporcionam ao longo da minha vida. Foram decisivos para conquistar tudo o que tenho hoje.

EPIGRAPH

"The mystery of human existence lies not in just staying alive,
but in finding something to live for."
Fyodor Dostoevsky (1821–1881)

ABSTRACT

This works presents a novel model based on deep learning for inflation forecasting, which is a daunting and unsolved problem in modern Macroeconomics. The challenges emerge due to the nonlinear and nonstationary behavior displayed by inflation in practice, diverging from the dynamics expected from the New Keynesian Phillips Curve. Consequently, conventional econometric models fail to deliver consistent and reliable forecasts, since they are not well-equipped to capture these complexities. In this context, deep learning presents itself as a promising approach, given its success when dealing with big data and nonlinearities. Illustrative examples abound in the fields of speech recognition, text interpretation, image processing, and financial time series modeling, among others. Astonishingly, despite its vast potential, no applications are available in the literature to investigate whether improvements in inflation forecasting can be obtained through deep learning. Thereby, as a contribution to the literature, this study proposes a hybrid deep learning model that merges Variational Autoencoders and Convolutional LSTM Networks to enhance the accuracy of inflation forecasts. The model estimation procedure employs state-of-the-art techniques to reduce overfitting, such as the addition of dropout and batch normalization layers in the model architecture. Through a public macroeconomic database that comprises 134 monthly US time series, the proposed model is compared against several popular econometric and machine learning benchmarks, including Ridge regression, LASSO regression, Random Forests, Bayesian methods, VECM, and multilayer perceptron. Using observations collected in the period ranging from January 1978 to December 2019, the empirical analysis corroborates the superiority of the model in terms of consistency and out-of-sample performance. The robustness of these findings is confirmed via cross-validation and simulations using different training, validation, and test samples.

Keywords: Deep Learning. Machine Learning. Inflation Forecasting. LSTM Networks. Convolutional Networks. Autoencoders.

RESUMO

Esse trabalho apresenta um modelo inovador baseado em *deep learning* para previsão de inflação, um problema desafiador e, até o momento, sem solução na Macroeconomia moderna. Os desafios emergem devido ao comportamento não linear e não estacionário exibido pela inflação na prática, divergindo da dinâmica esperada a partir da Curva de Phillips Neokeynesiana. Consequentemente, modelos econométricos convencionais se mostram incapazes de produzir previsões críveis e consistentes, pois não possuem a flexibilidade necessária para capturar essas complexidades. Nesse contexto, *deep learning* se apresenta como uma abordagem promissora, dado o seu sucesso no tratamento de dados não lineares e abundantes (*big data*). Exemplos ilustrativos são encontratos nos campos de reconhecimento de fala, interpretação textual, processamento de imagens e modelagem de séries temporais financeiras, entre outros. Surpreendentemente, apesar de seu potencial, não há aplicações de deep learning ao problema descrito para investigar se há a possibilidade de aprimorar previsões de inflação através dessa técnica. Portanto, como contribuição à literatura, esse estudo propõe um modelo de *deep learning* híbrido que combina *Autoencoders* Variacionais com Redes LSTM Convolucionais para ampliar a acurácia das previsões de inflação. O procedimento de estimação do modelo emprega técnicas do estado-da-arte para reduzir a probabilidade de *overfitting*, tais como a adição de camadas de *dropout* e *batch normalization* à arquitetura do modelo. Através de um banco de dados macroeconômicos públicos composto por 134 séries temporais mensais da economia estadunidense, o modelo proposto é comparado contra populares benchmarks econométricos e de *machine learning*, incluindo a regressão Ridge, a regressão LASSO, *Random Forests*, métodos Bayesianos, VECM, e o perceptron de múltiplas camadas. Usando observações coletadas no período que se estende de janeiro de 1978 até dezembro de 2019, a análise empírica corrobora a superioridade do modelo em termos de consistência e desempenho fora da amostra. A robustez das conclusões é confirmada mediante *cross-validation* e simulações usando diferentes amostras de treino, validação e teste.

Palavras-chave: Deep Learning. Machine Learning. Previsão de Inflação. Redes LSTM. Redes Convolucionais. Autoencoders.

EXECUTIVE SUMMARY

The study of inflation and mathematical models to explain its dynamics and generate forecasts is a recurring topic in modern Macroeconomics and Econometrics. Its relevance stems from the role played by inflation in many practical contexts. For instance, when implementing the monetary policy, central banks rely on inflation forecasts, among other inputs, to determine the optimal interest rate. Besides, firms and households savings and consumption are influenced by expected inflation. Furthermore, inflation is also crucial for financial institutions when calibrating credit spreads.

Despite the importance of inflation forecasting in realistic contexts, there is no consensus regarding which is the best econometric approach to conceive reliable and useful predictions. The challenges emerge from the complex empirical properties of observed inflation time series, marked by nonstationarity and nonlinearities. Standard econometric models usually do not have enough flexibility to reproduce these properties, since they are built on many simplifying assumptions incompatible with empirical evidence.

As illustration, it is worth mentioning the New Keynesian Phillips Curve (NKPC). Although supported by solid theoretical foundations and commonly employed in practice to investigate the behavior of inflation, the original NKPC usually displays lackluster performance in practice due to its linearity and constant coefficients. Many studies examined in this dissertation confirm the inaccurate forecasts produced by the NKPC and provide multiple explanations for the nonlinearities in inflation time series, such as downward nominal wage rigidities due to the labor rights faced when companies attempt to implement wage cuts.

Thereby, the main objective of this works is to investigate whether deep learning methods can increase the out-of-sample accuracy of inflation forecasts with respect to the conventional models typically encountered in the literature. The selection of deep learning, which is inserted into the set of machine learning algorithms, to overcome the limitation of previous approaches is justified by the promising results reported by researchers and practitioners in many contexts. Indeed, many authors have successfully applied deep learning not only for time series modeling, but also for complex tasks, such as pattern classification, speech recognition, and image processing. In these problems, data is typically nonlinear and nonstationary, similarly to inflation.

Surprisingly, until now, in spite of the potential of deep learning, applications to inflation modeling are somewhat scarce. The literature review unveils that Medeiros et al. (2019) were among the first to survey and carefully examine several machine learning models, evaluating their performance in forecasting inflation. Previous authors have focused exclusively on assessing individual models against straightforward benchmarks. Briefly, the results provided by Medeiros et al. (2019) demonstrate that machine learning models are able to consistently beat univariate econometric models, such as random walk and autoregressive ones, with gains as large as 30% in terms of mean squared errors. The superiority of machine learning is verified in multiple subsamples of the data collected and is robust in real-time experiments. These conclusions encourage this dissertation, given that deep learning is a subset of machine learning.

Hence, this works presents a novel model based on deep learning for inflation forecasting. More specifically, it is proposed a model merging Variational Autoencoders (VAE) and Convolutional LSTM (ConvLSTM) networks, two architectures that have been successfully applied to time series modeling. In that sense, the VAE acts as a dimension reduction tool, analogously to a nonlinear PCA, while the ConvLSTM forecasts inflation using the transformed inputs. The model estimation employs state-of-the-art techniques to prevent overfitting, including dropout and batch normalization layers in the architecture.

The performance of the model is compared against a wide set of benchmarks, which include popular econometric and machine learning models commonly found in the literature, including Ridge regression, LASSO regression, Random Forests, Bayesian methods, VECM, and multilayer perceptron. With this aim, the model and its benchmarks are adjusted through a public macroeconomic database comprising 134 monthly US time series. The period analyzed ranges from January 1978 to December 2019, covering several economic cycles and, thus, providing a fertile series of observations to test and compare the out-of-sample performance of the proposed model.

Briefly, the empirical results corroborate the superiority of the ConvLSTM coupled with VAE. This combination of networks provides the lowest median MSE across simulations and consistently outperforms its benchmarks. The robustness of these findings is confirmed via cross-validation and simulations using different training, validation, and test samples. The same conclusion is also obtained using distinct performance metrics, such as MAE, and is sustained in multiple forecasting horizons.

Additionally, these finds suggest that similar accomplishments may be attained using deep learning for modeling and forecasting other macroeconomic variables. Indeed, the factors that induce nonlinearities in inflation also influence other macroeconomic time series. Therefore, further studies may be carried out to inspect the application of deep learning models in these situations, comparing against alternatives available in the pertaining literature.

Finally, it should be emphasized that nonlinearities seem to be pervasive in the macroeconomic dataset contemplated in this works. This conclusion emerges from the fact that the VAE network managed to improve the performance of the ConvLSTM model, while linear factor models displayed mediocre out-of-sample accuracy. Jointly, these findings suggest that a nonlinear factor structure explain the variations of these macroeconomic variables, in agreement with similar studies using related datasets. Naturally, a thorough examination of this aspect would be pertinent for future projects, providing major contributions to the literature.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

In modern macroeconomics, the relevance of inflation forecasting cannot be overstated, given its prominent role in many practical situations. For instance, the estimation of DSGE models, which are pervasively employed by central banks in their decisions regarding monetary policy, requires a thorough understanding of inflation dynamics, without which it becomes unfeasible to derive links between this and other macroeconomic variables and, thus, make accurate predictions. Additionally, inflation forecasts are crucial for many firms when assessing the profitability of long-term investments. Finally, banks and households also rely on such analysis when celebrating contracts set in nominal values, such as debts.

As the pertaining literature shows, forecasting inflation is challenging and no consensus exists regarding which econometric approach is superior; see Faust and Wright (2013) and Rudd and Whelan (2007) for a comprehensive discussion. Undoubtedly, improving upon simple univariate econometric models is daunting due to several factors. The main hindrance lies on the nonlinear dynamics displayed by inflation in practice, undermining the use of the standard linear Phillips Curve, despite its theoretical appeal. Evidence is reported by Kumar and Orrenius (2016) and Zhang (2017), among others. This behavior explains the challenges, since significant improvements in forecasting can only be attained by capturing this dynamics via an appropriate econometric model designed to replicate such unconventional interactions between variables, which is far from trivial.

With respect to the nonlinearities, many sources have been identified in the literature. As shown by Daly and Hobijn (2014), given that firms virtually cannot reduce the nominal wages of their employees, the interaction between inflation and wages becomes nonlinear in face of downward nominal rigidities. Additionally, the zero lower bound for interest rates also induces nonlinear relationships between inflation, interest rates and other variables, as demonstrated by Grauwe and Ji (2019). Another source emerges from the degree of economic uncertainty, for it increases the option value of postponing decisions, as argued by Bloom (2009). Finally, Medeiros et al. (2019) observe that, if it is expensive to dismiss workers, then hiring, which affects inflation, should be a nonlinear function of uncertainty.

Further obstacles exist. Indisputably, they may stem from the choice of variables which can be systematically used for prediction, yielding reliable out-of-sample forecasts. In the era of big data, multiple options are promptly available. Without solid models and criteria to filter these variables, one becomes prone to data mining biases and overfitting. As illustration, in an effort to compile and standardize macroeconomic time series for academic research, McCracken and Ng (2016) provide a monthly database comprised by more than 100 variables that can be neatly applied to forecast inflation, as Medeiros et al. (2019) have demonstrated. However, it is conceivable that not all of these variables are necessary for out-of-sample prediction. If this is the case, inputting them without selection criteria may inevitably compromise the parsimony of the model and introduce noise. Hence, techniques for dimension reduction and denoising are required to address these issues.

In an attempt to tackle the aforementioned obstacles, the main objective of this work is to investigate whether deep learning methods can generate more accurate out-of-sample inflation forecasts than standard models reviewed in the related literature. The selection of deep learning, a type of machine learning, to overcome the limitations of previous approaches reflects the encouraging findings regarding its employment in real situations where nonlinearities are ubiquitous; see Goodfellow, Bengio and Courville (2016) for details. Indeed, many authors have successfully applied deep learning not only for time series modeling, but also for complex tasks, such as pattern classification (LEROUGE et al., 2015), speech recognition (LI; WU, 2015), and image processing (KIM et al., 2016), to list a few; see Liu et al. (2017) for additional examples. In these contexts, nonlinear features arise naturally, and deep learning has been particularly effective when dealing with them.

Curiously, until now, despite the potential of deep learning, applications to inflation modeling are somewhat scarce. The literature review carried out unveils that Medeiros et al. (2019) were among the first to survey and carefully examine several machine learning models, evaluating their performance in forecasting inflation. Previously, authors such as McAdam and McNelis (2005), Choudhary and Haider (2012), and Garcia, Medeiros and Vasconcelos (2017) have focused solely on assessing individual models against straightforward benchmarks. For this reason, the work of Medeiros et al. (2019) is among the main references for this dissertation. Briefly, the authors show that it is possible to consistently beat univariate models, such as random walk and autoregressive ones, with gains as large as 30% in terms of mean squared errors. The superiority of machine learning is verified in multiple subsamples of the data collected and is robust in real-time experiments.

Despite the significant contributions provided by the authors, there are opportunities for further research. Primordially, it must be highlighted that there is no consensus regarding the superiority of nonlinear models. For instance, Álvarez-Díaz and Gupta (2016) argue that accounting for nonlinearities does not necessarily provide statistical gains when forecasting the US CPI. Meanwhile, Ülke et al. (2018) conclude that, although machine learning models are more accurate under certain conditions, more parsimonious time series models, such as the ARDL (Autoregressive Distributed Lag), may perform better in some scenarios. Therefore, this topic demands additional academic scrutiny.

Another aspect to be discussed is the variety of machine learning methods available in the literature, including those belonging to the deep learning category. In that sense, Medeiros et al. (2019) seem to have focused on well-established models, but recent advances in deep learning have brought forth a myriad of promising alternatives for time series modeling, which are yet to be meticulously tested and appraised in the context of forecasting macroeconomic variables. For demonstration, multiple examples, with positive results, can be found in the surveys conducted by Atsalakis and Valavanis (2009), Ahmed et al. (2010), Längkvist, Karlsson and Loutfi (2017), and Tkáč and Verner (2016). Thus, a study of more modern approaches and their application for inflation forecasting is justified.

Before proceeding, it is paramount to establish the differences between deep learning and machine learning. Succinctly, the former distinguishes itself by the flexibility and predictive power that stem from the depth of the architecture of deep learning algorithms. As Goodfellow, Bengio and Courville (2016) emphasize, they achieve higher performance by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler ones, and more abstract representations computed in terms of less abstract ones. Accordingly, deep learning may be seem as one of many forms of machine learning.

For the purpose of illustrating the variety of architectures, it is worth mentioning LSTM (Long Short-Term Memory) networks, a versatile deep learning model that has been successfully applied in sequential processing such as textual interpretation. Introduced by Hochreiter and Schmidhuber (1997) and extended by Gers *et al.* (2000), Graves *et al.* (2008), and Cho *et al.* (2014), LSTM networks form a special subset of RNN (Recurrent Neural Networks) whose architecture is well-adapted to capture temporal dependencies in data. This feature explains their power for text processing, since, in such applications, the meaning of a word is not unconditional, depending on previous and upcoming words in a given sentence. Greff *et al.* (2016) present other uses involving representative tasks, such as handwriting recognition and polyphonic music modeling.

Analogously, in time series modeling, the dependent variable is assumed to be explained by its past values and other independent variables. Hence, LSTM networks might be suitable candidates to model this type of dependency in time series, justifying their use for forecasting inflation and, thus, explaining why they are one of the models examined in the present work. More precisely, a particular variation of LSTM network introduced by Shi et al. (2015), called Convolutional LSTM network, or ConvLSTM for short, is the focus due to its desirable flexibility for time series modeling and ability to extract spatiotemporal features from the inputs. Here, Bao, Yue and Rao (2017), Essien and Giannetti (2019), Fischer and Krauss (2018), Shi et al. (2015), and Wang, Qi and Liu (2019) become valuable references, demonstrating the uses of LSTM and ConvLSTM networks for time series, which can be immediately adapted for inflation with minor adjustments.

Furthermore, deep learning methods can also be used for dimension reduction and denoising. An example is the autoencoder, which is a type of neural network trained in an unsupervised manner for data encoding and decoding. When applied to time series, it can be seen as roughly equivalent to a nonlinear version of Principal Component Analysis (PCA). However, as Wang, Yao and Zhao (2016) argue, unlike PCA, autoencoders can also detect repetitive structure in the data. In various domains, these networks have achieved positive results, outperforming alternative dimension reduction techniques; see Wang, Yao and Zhao (2016) and Gu, Kelly and Xiu (2020).

Therefore, the contributions of this dissertation are threefold. First and foremost, by scrutinizing deep learning methods and applications in inflation forecasting, there is an opportunity to advance the literature related to the use of machine learning for macroeconomic prediction, which is a burgeoning and ever-changing field of research. In particular, it is possible to assess the potential of LSTM networks and their extensions, namely ConvLSTM, in modeling macroeconomic time series through the development of a novel deep learning model. For the preceding reasons, it is plausible to believe that deep learning could yield compelling results in several contexts that demand forecasting macroeconomic variables in general, without constraining itself to inflation.

Second, it is expected that, by using deep learning to model inflation, some knowledge regarding the sources of nonlinearities within this variable can be acquired. As previously contended, there is still no consensus with respect to the statistical gains of incorporating nonlinearities in models used for inflation forecasting. Actually, through the application of autoencoders, it becomes feasible to inspect the factor structure in the macroeconomic dataset to diagnosis whether nonlinear interactions are existent. Consequently, the present research may shed some light in these questions as well.

Finally, by examining denoising and dimension reduction techniques, this work intends to measure the value of data preprocessing for macroeconomic forecasting. In the era of big data, when multiple datasets are effortlessly available, data preparation becomes prominent so as to avoid data mining biases. Bernanke and Boivin (2003) coined the term *data-rich environment* precisely to describe these situations where both the number of variables and the length of time series are large and close to each other. It is reasonable to believe that, for macroeconomic time series, which are typically observed in low frequency, generating few observations, this topic is critical.

The next chapters are organized as follows. In chapter 2, a thorough literature review on inflation dynamics, econometric models, machine learning, and deep learning is put forward, covering the fundamental aspects required to understand the challenges involved in forecasting inflation, the origins of nonlinearities, and the reasons why deep learning may be effective at dealing with these complexities.

In chapter 3, the model developed for forecasting inflation is introduced and discussed. With this objective, convolutional LSTM networks and variational autoencoders are explored and techniques for merging these networks into a single architecture for time series modeling are presented and advocated.

Moreover, in chapter 4, the data employed when adjusting the models previously discussed and the methodology on which this work is built is explained. In that sense, the sources of the macroeconomic time series used for model estimation are set forth and the techniques and tools used in the empirical analysis are scrutinized. Some subsections are also devoted to explain the configuration and optimization of the neural networks fitted to the aforementioned data.

In the sequence, chapter 5 encompasses the results of the empirical analysis, which are detailed and examined. Briefly, this section summarizes the accuracy of the proposed model and of the selected benchmarks in the designed experiments. Several performance metrics are computed for this purpose, allowing the comparison via out-of-sample data. Notably, the model proposed surpasses its benchmarks, corroborating that deep learning can effectively capture the stochastic process that governs inflation in practice.

Finally, in chapter 6, the main conclusions of this research are reviewed. Possible extensions of the current work are also suggested, laying ground for future projects in the field.

2 LITERATURE REVIEW

In this section, the literature review regarding the main topics covered in this work is carried out. With this purpose, initially a discussion on inflation forecasting and the New Keynesian Phillips Curve is presented. Subsequently, usual econometric and machine learning models with applications in Economics are introduced and debated. Finally, the literature on deep learning models is scrutinized, with special attention to the architectures that underpin the model advocated herein for inflation forecasting.

2.1 Inflation Forecasting

In the past decades, inflation forecasting has attracted the attention of both practitioners and researchers, conceiving an extensive and prolific literature on the subject. The focus on this econometric problem is explained by the prominent role played by inflation in many real situations. In spite of the importance of forecasting inflation accurately, improving upon simple models has been a daunting and frustrating task, as argued by Medeiros et al. (2019). Furthermore, the literature review shows that there is no consensus regarding which econometric model is more appropriate, yielding superior and consistent forecasts.

With the aim of presenting the recent advances and analyzing the current status of the academic research, we begin by examining the New Keynesian Phillips Curve (NKPC), an usual starting point justified by its theoretical appeal. By scrutinizing the NKPC, it is possible to demonstrate its limitations and the reasons explaining why forecasting inflation is complex, demanding the employment of sophisticated econometric methods. Later, alternatives are suggested, and deep learning is introduced as a promising solution.

2.1.1 New Keynesian Phillips Curve (NKPC)

As a theoretical framework for inflation dynamics, the New Keynesian Phillips Curve (NKPC) is often resorted to as foundation and justification for some of the earlier econometric models for forecasting inflation. Following Gali and Gertler (1999), Mavroeidis, Plagborg-Møller and Stock (2014), and Gali (2015), it is assumed a small economy in which monopolistic competitive firms face some type of constraints on price adjustment.

For the sake of simplification of the pricing rules and the consequent aggregation of prices histories, the mechanism proposed by Calvo (1983) is adopted. Hence, it is assumed that, in any given period, each firm has a fixed probability $1 - \theta$ of updating prices during that period, where θ is independent of the time elapsed since the last adjustment. Thus, with probability θ, prices will be kept unchanged.

Now, it is imposed that firms seek to maximize their profits and are identical, except for the differentiated product they produce and for their pricing history. Additionally, we assume that firms face a conventional constant price elasticity of demand curve for their products. In that context, it is possible to show that the aggregate price level p_t evolves as a convex combination of the lagged price level p_{t-1} and the optimal reset price p_t^* (i.e. the price chosen by firms that are able to adjust prices at t), as follows:

$$p_t = \theta p_{t-1} + (1-\theta)p_t^* \tag{2.1}$$

Under these conditions, let mc_t^n be the firm's nominal marginal cost at t (as a percentage deviation from the steady state) and let β denote a subjective discount factor. Then the optimal reset price may be stated as:

$$p_t^* = (1-\beta\theta)\sum_{k=0}^{\infty}(\beta\theta)^k E_t\left[mc_{t+k}^n\right] \tag{2.2}$$

where E_t denotes the expected value conditional on the information available at instant t. Therefore, in setting its price at t, a firm takes into account the expected future path of nominal marginal cost, given the likelihood that its price may be kept fixed for several periods ahead.

Now, let π_t denote the inflation rate at t, and mc_t the percent deviation of the firm's real marginal cost from its steady state value. Using the previous equations, it is manageable to derive an inflation equation of the form:

$$\pi_t = \beta E_t[\pi_{t+1}] + \lambda mc_t = \lambda\sum_{k=0}^{\infty}\beta^k E_t[mc_{t+k}] \tag{2.3}$$

where:

$$\lambda = \frac{(1-\theta)(1-\theta\beta)}{\theta} \tag{2.4}$$

implying that inflation should equal a discounted stream of expected future marginal costs.

Next, it is possible to show that, adopting certain assumptions:

$$mc_t = \kappa(y_t - y_t^*) \tag{2.5}$$

where $(y_t - y_t^*)$ is the output gap and κ is the output elasticity of marginal cost. Combining this formula with previous results, one obtains the classic NKPC:

$$\pi_t = \beta E_t[\pi_{t+1}] + \lambda\kappa(y_t - y_t^*) \tag{2.6}$$

Estimation of the coefficients of this NKPC can be done as follows. Since, under rational expectations, the error in the forecast of π_{t+1} is independent of the filtration $\mathcal{F}_t$, then:

$$E_t[(\pi_t - \lambda mc_t - \beta\pi_{t+1})z_t] = 0 \tag{2.7}$$

where z_t is a vector of variables dated t and earlier, thus orthogonal to the inflation surprise in period $t+1$. This allows the estimation via GMM (Generalized Method of Moments); see Dennis (2006).

A natural extension of the previous NKPC involves the assumption of indexation. That is, every period, some firms adopt a backward-looking rule-of-thumb to set prices. This approach is developed by Gali and Gertler (1999) assuming that only a fraction of firms adopt a Calvo-based, forward-looking rule to set prices. The remaining ones instead use a simple rule-of-thumb that is based on the recent history of aggregate price behavior. Hence, there is a certain degree of indexation in the economy, which is related to τ, the fraction of inflation transferred to final prices by backward-looking firms. In this setting, it is possible to demonstrate that the NKPC becomes:

$$\pi_t = \frac{1}{1+\theta\beta\tau}\left(\tau\pi_{t-1} + \beta E_t\left[\pi_{t+1}\right]\right) + \frac{(1-\theta)(1-\theta\beta)}{\theta(1+\theta\beta\tau)}\kappa(y_t - y_t^*) \tag{2.8}$$

Thus, in this extension, current inflation depends not only on the output gap and on inflation expectations, but also on past inflation.

2.1.2 Sources of Nonlinearities in Inflation Dynamics

In the previous subsection, the main assumptions of the NKPC were disclosed. In essence, the derivation of the NKPC reveals that its original version and most variations are a mixture of linear forward- and backward-looking models. That is, they assume the form (MAVROEIDIS, 2005):

$$\pi_t = \beta E(\pi_{t+1}|\mathcal{F}_t) + \gamma\pi_{t-1} + d_t \tag{2.9}$$

where π_t is a decision variable (i.e. inflation), d_t is a "driving" or "forcing" variable, usually deemed exogenous, and $E(\pi_{t+1}|\mathcal{F}_t)$ is the expectation of π_{t+1} conditional on the filtration at time t. As argued by Mavroeidis (2005), the popularity of such models arises from the fact that they reproduce a forward-looking economic decision process, addressing the so-called Lucas Critique.

In practice, despite the appeal of this family of models, many issues have been identified, explaining why the NKPC displays weak performance when adjusted using real data, as shown by Atkeson and Ohanian (2001) and others. Indeed, Atkeson and Ohanian (2001) argue that the likelihood of accurately predicting changes in inflation rate via the outcomes of these models produces results statistically equivalent to using a coin flip for forecasting.

With respect to the issues highlighted in the literature, empirical studies have reported that a linear NKPC with constant coefficients is unsuitable due to the presence of a time-varying slope. As illustration, Blanchard (2016) argues that, in US, the slope has substantially declined in the past decades, which is unfeasible in the classic NKPC framework. Such limitations pose several hindrances, for the existence of nonlinearities in the Phillips curve has material implications for monetary policy, as shown by Schaling (2004).

Moreover, rigidities may induce nonlinearities in inflation, creating more complex dynamics. Indeed, Daly and Hobijn (2014) develop a model of monetary policy with downward nominal wage rigidities stemming from the fact that wage cuts are rare and, consequently, log wage changes follow non-normal distributions. They demonstrate that both the slope and the curvature of the Phillips curve vary according to the level of inflation and the intensity of such rigidities. This is verified both in the short and long-run curves, which, again, violates the traditional models for the NKPC. Their findings are consistent with evidence suggesting that, in certain periods when unemployment declines, wage growth may be modest or even nonexistent.

Supplementary material is provided by Correa and Minella (2010), who examine the presence of nonlinear mechanisms of pass-through from the exchange rate to inflation in Brazil. In particular, they estimate a Phillips Curve with thresholds to replicate this behavior. The results suggest that the short-run pass-through is higher when the economy is growing faster, when the exchange rate depreciates above some threshold and when exchange rate volatility is lower. In a similar fashion, using Markov-switching autoregressive models, Binner et al. (2006) conclude that these regime-switching systems capture nonlinearities in inflation data and improve forecasting performance.

Furthermore, Ball and Mazumder (2011) present a puzzle that arises when Phillips curves are estimated over 1960-2007: inflation should have fallen by more than it did. The puzzle is resolved with two modifications of the Phillips curve stemming from theories of costly price adjustment: (a) core inflation is measured using the weighted median of consumer price inflation rates across industries; and (b) the slope of the Phillips curve is allowed to vary with the level and variance of inflation. The authors also conclude that the Great Recession provides fresh evidence against the New Keynesian Phillips curve with rational expectations.

Finally, as another piece of evidence in favor of nonlinearities, Aruoba, Bocola and Schorfheide (2017) develop a new class of time series models to identify nonlinearities in data and evaluate DSGE models. They estimate a DSGE model with asymmetric wage and price adjustment costs, due to which it becomes possible to match the nonlinear inflation and wage dynamics observed in practice.

Other aspects of the NKPC must also be restructured so as to address empirical observations. For instance, Coibion and Gorodnichenko (2015) propose that an expectation-augmented Phillips curve that proxies firms' inflation expectations by household expectations is capable of explaining the missing disinflation during the Great Recession. The authors identify information rigidity in US and international data and document evidence of state-dependence in the expectation formation process.

Naturally, there is evidence against the adoption of nonlinear models as well. For instance, Álvarez-Díaz and Gupta (2016) argue that accounting for nonlinearity does not necessarily provide statistical gains when forecasting US CPI. Meanwhile, Ülke et al. (2018) conclude that, although machine learning models are more accurate under certain conditions, more parsimonious time series models, such as the ARDL model, may outperform in some scenarios.

However, even if one assumes nonlinear interactions do not exist or are unnecessary, the fact that coefficients change through time is sufficient to demand more sophisticated models for inflation forecasting. Also, it is worth noting that inflation displays heavy tails (MONACHE; PETRELLA, 2017) and time-varying volatility (CLARK; DOH, 2014), amplifying the challenges discussed in this subsection.

2.2 Conventional Econometric Models

In this section, we begin exploring econometric models commonly used for macroeconomic time series. The objective here is to introduce these models so as to justify the selection of some of them as benchmarks for the deep learning model developed for inflation forecasting. Since they are merely accessory and not the focus of this work, solely a brief introduction is provided herein. The interested reader may seek Hastie, Tibshirani and Friedman (2008), James et al. (2013), Tsay (2010), and other references cited for further details.

2.2.1 ARFIMA

The ARFIMA model was introduced by Hosking (1981). It is commonly used for time series that display long-term memory, defined as variables whose autocorrelation function (ACF) exhibits polynomial decay in time, in opposition to an exponential decay implicit in SARIMA models, for instance. More formally, these series are characterized by a fractional degree of integration.

Therefore, ARFIMA is based on fractional differencing. Following Tsay (2010), the model is defined as:

$$(1-B)^d x_t = a_t, \quad -0.5 < d < 0.5 \tag{2.10}$$

where x_t is a given time series and a_t is a white noise. As shown in Tsay (2010), if $d < 0.5$, then x_t is a weakly stationary process. If $d > -0.5$, then x_t is invertible and has an infinite AR representation. For $-0.5 < d < 0.5$, the ACF of x_t is:

$$\rho_k = \frac{d(1+d)\cdots(k-1+d)}{(1-d)(2-d)\cdots(k-d)}, \quad k = 1, 2, \ldots \tag{2.11}$$

Therefore:

$$\rho_k \approx \frac{(-d)!}{(d-1)!} k^{2d-1} \quad \text{as} \quad k \to \infty \tag{2.12}$$

meaning that the ACF decays at a polynomial rate.

An application of ARFIMA models for inflation forecasting is provided by Baillie, Chung and Tieslay (1996). The authors implement a novel procedure to obtain approximate maximum likelihood estimates of ARFIMA-GARCH process. The model is employed on the analysis of monthly inflation rates for ten different countries, and the authors find strong evidence of long memory with mean-reverting behavior for almost all countries in the sample considered, with exception of Japan.

2.2.2 GARCH

Developed by Bollerslev (1986), the GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model is a widespread approach for time series marked by time-varying volatility. The GARCH(m, n) process is described by the following system of equations (TSAY, 2010):

$$r_t = \mu_t + \varepsilon_t, \quad \varepsilon_t \sim N(0, \sigma_t^2) \tag{2.13}$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{m} \beta_i \varepsilon_{t-i}^2 + \sum_{j=1}^{n} \gamma_j \sigma_{t-j}^2 \tag{2.14}$$

$$\sum_{k=1}^{max(m,n)} (\beta_k + \gamma_k) < 1 \tag{2.15}$$

$$\alpha_0 > 0, \quad \beta_i \geq 0, \quad \gamma_j \geq 0 \tag{2.16}$$

where r_t is a given time-series, μ_t is the mean process, ε_t is a sequence of i.i.d. random variables with mean 0 and variance 1, σ_t^2 is the variance of r_t, and α_0, β_i, $i = 1, \ldots, m$, and γ_j, $j = 1, \ldots, n$, are coefficients to be estimated.

An application of GARCH models in the context of inflation forecasting is provided by Kontonikas (2004). The paper investigates the connection between inflation and uncertainty and the effects of inflation targeting using British data over the period of 1972–2002. The results suggest a positive relationship between past inflation and current uncertainty, and indicate that it would be beneficial for countries with implicit targeting to adopt formal inflation targets.

2.2.3 VAR and Cointegration

According to Lütkepohl (2005), a VAR(p) model (Vector Autoregressive Model of order p) is defined as:

$$y_t = \mu + A_1 y_{t-1} + \cdots + A_p y_{t-p} + u_t, \quad t = 0, 1, 2, \ldots \tag{2.17}$$

where y_t is a $k \times 1$ random vector, the A_i are fixed $k \times k$ coefficient matrices, μ is a fixed $k \times 1$ vector of intercept terms allowing for the possibility of a nonzero mean $E(y_t)$. Finally, u_t is a k-dimensional white noise or innovation process. The covariance matrix $E(u_t u_t^t) = \Sigma_u$ is assumed to be nonsingular.

Under general conditions, for a k-dimensional unit-root nonstationary time series, *cointegration* exists if there are less than k unit roots in the system (TSAY, 2010). Let h be the number of unit roots in the k-dimensional series y_t. Cointegration exists if $0 < h < k$, and the quantity $k - h$ is called the number of *cointegrating factors*. Alternatively, the number of cointegrating factors is the number of different linear combinations, called *cointegrating vectors*, that are unit-root stationary.

An insightful reference of a cointegration model applied to inflation forecasting is provided by Cologni and Manera (2008). The authors propose a structural cointegrated VAR model for G-7 countries so as to investigate the direct impacts of oil price shocks on output and prices, as well as the effects and reactions of monetary variables to external shocks. The study is motivated by the fact that sharp increases in oil prices are generally perceived as a major contributor to business cycle asymmetries.

The structural cointegrated VAR model starts by considering the following reduced-form vector error correction model (VECM) with k variables (COLOGNI; MANERA, 2008):

$$\Delta y_t = \Pi y_{t-1} + \Gamma_1 \Delta y_{t-1} + \ldots + \Gamma_{p-1} \Delta y_{t-p+1} + u_t \tag{2.18}$$

where y_t, $t = 1, \ldots, T$, is a $k \times 1$ vector of time series and Π is a $k \times k$ matrix which has reduced rank $r < k$ and can be decomposed as $\Pi = \alpha\beta^t$. Matrices α and β have dimension $k \times r$ and contain the loading coefficients and the cointegration vectors, respectively. Moreover, Γ_i, $i = 1, \ldots, p-1$, are $k \times k$ short-run coefficients matrices, and u_t is a white noise error vector with zero mean and nonsingular covariance matrix Σ_u.

The next step comprises the identification of structural innovations, which induce informative responses of the variables in the system. In particular, the effects of the fundamental shocks ε_t on the system variables y_t can be described by the following structural specification:

$$A\Delta y_t = \Psi y_{t-1} + \Lambda_1 \Delta y_{t-1} + \ldots + \Lambda_{p-1} \Delta y_{t-p+1} + \nu_t \tag{2.19}$$

where Ψ and Λ_i, $i = 1, \ldots, p-1$, are coefficients and the $k \times 1$ vector of structural disturbances ν_t has zero mean and covariance matrix Σ_ν. The responses to the economic shocks ϵ_t, forecast errors u_t must be linked to the structural shocks ε_t. Multiplying both sides of the previous equation by A^{-1} gives the reduced-form of equation Equation 2.18, where $\Pi = A^{-1}\Psi$ and $\Psi_i = A^{-1}\Lambda_i$, $i = 1, 2, \ldots, p-1$ and:

$$u_t = A^{-1}\epsilon_t = B\epsilon_t \tag{2.20}$$

Equation 2.20 relates the reduced-form disturbance u_t to the underlying structural shocks ϵ_t. As put by Cologni and Manera (2008), the analysis of the effects of the underlying structural shocks on the system variables requires to recover the k^2 elements of B. A set of restrictions arising from the underlying economic theory may be used, and it is possible to write:

$$\Sigma_u = E\left[u_t u_t^t\right] = BE\left[\epsilon_t \epsilon_t^t\right] = B\Sigma_\epsilon B^t \tag{2.21}$$

For a unique specification of the k^2 elements of B, at least $k(k-1)/2$ further restrictions are necessary.

2.2.4 Regime-Switching Models

Regime-switching models are popular for nonstationary time series. Here, two common techniques are described: (1) Markov chain models; and (2) threshold autoregressive models.

2.2.4.1 Markov Chain Models

In their seminal work, Hamilton (1989) proposes a tractable approach to modeling changes in regime. The idea is to link the parameters of an autoregression to the outcomes of a discrete-state Markov process. By introducing this flexibility, it becomes possible to model shifts in the mean of a nonstationary series, for instance.

Following Hamilton (1989) and Tsay (2010), a time series x_t is governed by a Markov switching autoregressive (MSA) model if it satisfies:

$$x_t = \begin{cases} c_1 + \sum_{i=1}^{p} \phi_{1,i} x_{t-i} + a_{1t}, & \text{if } s_t = 1 \\ c_2 + \sum_{i=1}^{p} \phi_{2,i} x_{t-i} + a_{2t}, & \text{if } s_t = 2 \end{cases} \tag{2.22}$$

where s_t assumes values in $\{1, 2\}$ and is a first-order Markov chain with transition probabilities:

$$p(s_t = 2 | s_{t-1} = 1) = p_1 \tag{2.23}$$

$$p(s_t = 1 | s_{t-1} = 2) = p_2 \tag{2.24}$$

The innovations a_{1t} and a_{2t} are sequences of i.i.d. random variables with mean zero and finite variance and are independent of each other. Hamilton (1989) shows an application of MSA models using GNP data to successfully identify business cycles.

2.2.4.2 Threshold Autoregressive Models

As established by Tsay (2010), a time series y_t is said to follow a k-regime Self-Exciting Threshold Autoregressive Model (SETAR) model with threshold variable y_{t-d} if it satisfies:

$$y_t = \phi_0^{(j)} + \phi_1^{(j)} x_{t-1} - \cdots - \phi_p^{(j)} x_{t-p} + a_t^{(j)}, \quad \gamma_{j-1} \leq y_{t-d} < \gamma_j \tag{2.25}$$

where k and d are positive integers, $j = 1, \ldots, k$, γ_i are real numbers such that $-\infty = \gamma_0 < \gamma_1 < \cdots < \gamma_{k-1} < \gamma_k = \infty$, the superscript (j) signifies the regime, and $a_t^{(j)}$ are i.i.d. sequences with mean 0 and variance σ_j^2 and are mutually independent for different j. The parameter d determines the delay and γ_j are the thresholds. An implicit assumption is that each regime has a distinct AR model, otherwise the number of regimes could be reduced.

Threshold models have had some success in modeling nonlinearities in inflation dynamics. Indeed, using monthly data over the period 1976–2002, Khadaroo (2005) detects significant nonlinearities in the inflation rates of India, Singapore, and South Africa. The author then estimates a two-regime SETAR model and observes material improvement over the corresponding linear AR model.

2.2.5 Factor Models

Following Stock and Watson (2016), *dynamic factor models* (DFM) represent the evolution of a vector of N observed time series, X_t, in terms of a reduced number of unobserved common factors evolving over time, plus uncorrelated disturbances representing measurement error and/or idiosyncratic dynamics of the individual series. Such model can be written in two ways: (1) the dynamic form, expressing the dependence of X_t on lags (and possibly leads) of the factors explicitly; or (2) the static form, representing those dynamics implicitly.

Formally, according to Stock and Watson (2016), the dynamic form of the DFM is expressed as:

$$X_t = \lambda(L) f_t + e_t \tag{2.26}$$

$$f_t = \Psi(L) f_{t-1} + \eta_t \tag{2.27}$$

where the lag polynomial matrices $\lambda(L)$ and $\Psi(L)$ are $N \times q$ and $q \times q$, respective, and η_t is a $q \times 1$ vector of serially uncorrelated mean-zero innovations to the factors. In general, e_t can be serially correlated, although it is assumed that $E\left(e_t \eta_{t-k}^t\right) = 0, \forall k$. It must be emphasized that, if e_t is serially correlated, then the preceding specification is incomplete, requiring a equation that describes the temporal evolution of e_t. The static form rewrites this set of equations in terms of r static factors F_t, instead of q dynamic factor f_t, where $r \geq q$; see Stock and Watson (2016) for further details.

The main advantage of DFM in macroeconometric applications is the ability to reduce the complex comovements of a potentially large number of observable variable into a small set of factors that explain most of the common fluctuations of all the series (STOCK; WATSON, 2016). In that sense, they are an example of a larger class of state-space or hidden Markov models, in which observable (measurable) variables are written in terms of unobserved (latent) ones. The latter evolve according to some lagged dynamics with finite dependence (STOCK; WATSON, 2016).

According to Bai and Ng (2008), *targeted factors* can be designed by targeting the predictors, offering an alternative approach for designing factor models. Through the method denominated *hard thresholding*, the authors propose a strategy to select the relevant predictors. The idea consists on initially identifying statistically significant predictors of the dependent variables and, subsequently, running PCA to extract the factors.

Another version is introduced by Bai and Ng (2009), who merge factor models with the boosting technique as a strategy for selecting the predictors in factor-augmented autoregressions. The authors also develop a stopping rule for boosting to prevent the model from being overfitted with estimated predictors. The empirical analysis corroborates the success of factor boosting.

An example of empirical application in Macroeconomics is provided by Gupta and Kabundi (2011), who employs large factor models that accommodate a wide set of 267 macroeconomic time series for forecasting the per capita growth rate, inflation, and the nominal short-term interest rate for South Africa. The model proposed outperforms alternatives such as VAR, Bayesian VAR and DSGE models.

2.3 Shrinkage Methods

In Statistics, *shrinkage* methods are used to reduce coefficient variance and improve out-of-sample performance in regression analysis. They encompass techniques that constrain or regularize coefficient estimates, or, equivalently, that *shrink* these estimates towards zero, improving goodness-of-fit (JAMES et al., 2013). In the next subsections, some shrinkage approaches that will be later used as benchmarks are discussed.

2.3.1 LASSO Regression

LASSO (Least Absolute Shrinkage and Selection Operator) regression is a method popularized by Tibshirani (1996). The motivation behind this approach is that, in many practical situations, mainly when dealing with large datasets, only a subset of the existing covariates is required to provide a satisfactory explanation of the variance of the independent variable.

Therefore, some coefficients in the regression are shrunk towards zero by adding a penalty to the loss function that grows with the size of each coefficient, forcing some of them to be exactly zero (HASTIE; TIBSHIRANI; FRIEDMAN, 2008). Thus, for a linear regression of the form:

$$y = X\beta + \varepsilon \tag{2.28}$$

The LASSO estimator becomes:

$$\hat{\beta} = \arg\min_{\beta} \left(\|y - X\beta\|_2^2 + \lambda \sum_{j=1}^{k} |\beta_j| \right) \tag{2.29}$$

where β is the $k \times 1$ vector of parameters, y is the vector of T observations of the dependent variable, X is a $T \times k$ matrix containing the independent variables, ε is a vector of residuals, and $\|\cdot\|_2$ denotes the Euclidean norm. In addition, λ is a tuning parameter that determines the relative importance between minimizing the sum of squared residuals and removing uninformative variables. Although λ can be predefined by the user, information criteria and cross-validation are common approaches for the estimation of this parameter.

Clearly, the penalty introduced assumes the form of a ℓ_1-norm, which makes the solution nonlinear in y and is more restrictive than the ℓ_2-norm used in the Ridge regression discussed subsequently. This imposes that some parameters of the regression become exactly equal to zero when the tuning parameter λ is sufficiently large. As implication, LASSO imposes an efficient variable selection mechanism in exchange of biasing the estimate $\hat{\beta}$ (consistency is preserved, however). The drawback is that there is no closed form expression for $\hat{\beta}$, requiring numerical solutions.

A Bayesian modification of the LASSO estimator was introduced by the seminal work by Park and Casella (2008) and extended by Hans (2009), among others. Initially, it should be noted that the standard form is by itself Bayesian, since, as demonstrated by Hastie, Tibshirani and Friedman (2008) and Park and Casella (2008), LASSO estimates for linear regression parameters can be interpreted as a Bayesian posterior mode estimate when the regression parameters have independent Laplace priors. That is, LASSO can be expressed under the Bayesian framework as:

$$p(y|\beta, \sigma^2, \tau) = N(y|X\beta, \sigma^2 I_n) \tag{2.30}$$

$$p(\beta|\lambda, \sigma^2) = \left(\frac{\lambda}{2\sigma}\right)^k e^{\lambda\sigma^{-1}\|\beta\|_1} \tag{2.31}$$

where $\|\cdot\|$ is the ℓ_1 norm and λ is a hyperparameter. Conditioning on σ^2 is crucial, because it guarantees a unimodal full posterior (PARK; CASELLA, 2008).

Seeking to extend this framework and incorporate a Gibbs sampler, Park and Casella (2008) suggest the following hierarchical representation of the full Bayesian model:

$$y|\mu, X, \beta, \sigma^2 \sim N(\mu 1_n + X\beta, \sigma^2 I_n) \tag{2.32}$$

$$\beta|\sigma^2, \tau_1^2, \ldots, \tau_k^2 \sim N_k(0_k, \sigma^2 D_\tau) \tag{2.33}$$

$$D_\tau = \text{diag}(\tau_1^2, \ldots, \tau_k^2) \tag{2.34}$$

$$\sigma^2, \tau_1^2, \ldots, \tau_p^2 \sim \pi(\sigma^2) d\sigma^2 \prod_{j=1}^{k} \frac{\lambda^2}{2} e^{-\lambda^2 \tau_j^2/2} d\tau_j^2 \tag{2.35}$$

$$\sigma^2, \tau_1^2, \ldots, \tau_k^2 > 0 \tag{2.36}$$

where μ is a hyperparameter which may be given an independent, flat prior, and 1_n is a vector of ones. An improper prior density $\pi(\sigma^2) = 1/\sigma^2$ or any inverse-gamma prior for σ^2 may be used to maintain conjugacy. Meanwhile, λ may be approximated iteratively throughout the execution of the Gibbs sampler or a prior may be selected. Park and Casella (2008) propose the following class of gamma priors with parameters r and δ:

$$\pi(\lambda^2) = \frac{\delta^r}{\Gamma(r)} \left(\lambda^2\right)^{r-1} e^{-\delta\lambda^2}, \quad r > 0, \delta > 0 \tag{2.37}$$

where $\Gamma(\cdot)$ is the Gamma function.

LASSO-based approaches for time series modeling have had success in macroeconomics. As a case in point, evaluating the predictive ability of multiple models for forecasting twenty important macroeconomic variables, Li and Chen (2014) show that combining forecasts from LASSO-based models with those from dynamic factor models can reduce the out-of-sample MSE. The authors also verify that LASSO-based models outperform dynamic factor ones in terms of out-of-sample performance. Ultimately, they conclude that LASSO can be useful for extracting information and formulating economically meaningful predictors.

2.3.2 Ridge Regression

Developed by Hoerl and Kennard (1970), the Ridge estimator for a linear regression is defined as follows:

$$\hat{\beta} = \arg\min_{\beta} \left(\|y - X\beta\|_2^2 + \lambda \sum_{j=1}^{k} \beta_j^2 \right) \tag{2.38}$$

where λ is a predefined shrinkage parameter chosen by the user. Mathematically, it corresponds to a special case of the Tikhonov regularization for ill-posed problems.

Unlike LASSO, whose advantage is covariate selection, the Ridge estimator attempts to regularize by shrinking parameters associated with redundant predictors without completely excluding them. Hence, at the cost of introducing some bias in the estimator of β, the Ridge estimator tackles multicollinearity, which emerges when there are many correlated variables in a linear regression model (HASTIE; TIBSHIRANI; FRIEDMAN, 2008).

Another advantage of Ridge regression is that the optimization problem formulated in the previous equation is always convex for $\lambda > 0$ and its solution can be expressed in closed-form as:

$$\hat{\beta} = (X^t X + \lambda I)^{-1} X^t y \tag{2.39}$$

where X^t denotes the transpose of X. As one can infer from the previous equation, similarly to the OLS, the solution is a linear function of y. The addition of the identity matrix to $X^t X$ ensures a single solution exists even if this product results in a singular matrix. Analogously to the LASSO estimator, the Ridge estimator adds a bias to the estimator $\hat{\beta}$ in order to achieve regularization. Still, consistency remains intact. Also, λ may also be set using cross-validation or information criteria, balancing goodness-of-fit with parsimony. A Bayesian version of Ridge also exists and can be implemented using the approach by Tipping (2001), for instance.

Exemplifying the application of Ridge estimators for financial and macroeconomic variables, Kim and Swanson (2014) empirically assess the predictive accuracy of a large group of models based on principal components and shrinkage methods, including Bayesian model averaging, bagging, boosting, least angle regression, Ridge, to name a few. The results confirm that model averaging does not dominate other well calibrated and specified models. In addition, merging factor / shrinkage methods often yields superior accuracy.

2.3.3 Elastic Net

Elastic Net is a generalization of LASSO and Ridge estimators, including both as particular cases (HASTIE; TIBSHIRANI; FRIEDMAN, 2008). Essentially, Elastic Net is a convex combination of ℓ_1- and ℓ_2-norm. Thus, it simultaneously regularizes and selects the most relevant variables. The estimator $\hat{\beta}$ is the solution of:

$$\hat{\beta} = \arg\min_{\beta} \left[\|y - X\beta\|_2^2 + \alpha\lambda \sum_{j=1}^{k} |\beta_j| + (1-\alpha)\lambda \sum_{j=1}^{k} \beta_j^2 \right] \tag{2.40}$$

The parameter $\alpha \in [0, 1]$ controls the weights given to each norm and, together with λ, can be found via cross-validation or minimization of information criteria over a grid of candidate values.

2.4 Ensemble Models

Ensemble learning is a strategy used to produce more accurate and stable models. Essentially, it involves building a prediction model by combining the strengths of a collection of simpler base models (HASTIE; TIBSHIRANI; FRIEDMAN, 2008).

2.4.1 Gradient Boosting and AdaBoost

According to Friedman (2002), *gradient boosting* builds additive regression models by sequentially fitting a simple parameterized function to current "pseudo"-residuals by least squares at each iteration. Mathematically, in a general function estimation problem, one wants to find a function $F^*(x)$ that maps a vector of explanatory variables x to a dependent variable y with the goal of minimizing the expected value of some predefined loss function L. That is:

$$F^*(x) = \arg\min_{F(x)} E_{x,y}\left[L(y, F(x))\right] \tag{2.41}$$

Boosting approximates $F^*(x)$ by an additive expansion of the form (FRIEDMAN, 2002):

$$F(x) = \sum_{m=0}^{M} \beta_m h(x; a_m) \tag{2.42}$$

where $h(x; a_m)$ are simple, parameterized functions in a_m. Both β_m and a_m are jointly fit to the training data in a forward "stage-wise" procedure, starting with an initial guess $F_0(x)$, and then, for $m = 1, 2, \ldots, M$:

$$(\beta_m, a_m) = \arg\min_{\beta,a} \sum_{i=1}^{N} L\left(y_i, F_{m-1}(x_i) + \beta h(x_i; a)\right) \tag{2.43}$$

where N is the number of observation in the sample collected and:

$$F_m(x) = F_{m-1}(x) + \beta_m h(x; a_m) \tag{2.44}$$

Gradient boosting approximately solves Equation 2.43 for an arbitrary differentiable loss L in two stages. First, the algorithm fits h by least squares:

$$a_m = \arg\min_{\rho,a} \sum_{i=1}^{N} \left[\hat{y}_{im} - \rho h(x_i; a)\right]^2 \tag{2.45}$$

where:

$$\hat{y}_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \tag{2.46}$$

Then, given $h(x; a_m)$, the optimal β_m can be determined:

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{N} L\left(y_i, F_{m-1}(x_i) + \beta h(x_i; a_m)\right) \tag{2.47}$$

This solving strategy replaces a nontrivial function optimization problem Equation 2.43 by one based on least squares, followed by a single parameter optimization in terms of the loss L (FRIEDMAN, 2002). *Adaptive boosting*, or *AdaBoost*, works in a similar manner, minimizing an exponential loss function and adjusting the parameters accordingly; see Collins, Schapire and Singer (2002).

2.4.2 Bagging

Boostrap aggregation, or *bagging*, is a general procedure for variance reduction in econometric models. The model was originally proposed by Breiman (1996), who advocated the combination of forecasts from several distinct models fitted for multiple bootstrap samples. Tests conducted on real and simulated data sets using classification and regression trees and other methods demonstrate that bagging can yield substantial accuracy gains.

Inspired by Hastie, Tibshirani and Friedman (2008), initially it is considered a regression problem where we fit a model to a training data $\boldsymbol{Z} = (x_1, y_1), \ldots, (x_n, y_n)$, yielding the prediction function $\hat{f}(x)$ for input x. The bagging technique averages this prediction over a collection of bootstrap samples, which contributes for variance reduction. In practice, for each bootstrap samples $\boldsymbol{Z}_b^*$, $b = 1, 2, \ldots, B$, the model is fitted and gives prediction $\hat{f}_b^*(x)$. The bagging estimate is then defined by:

$$\hat{f}_{bagging}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b^*(x) \tag{2.48}$$

The expression above is a Monte Carlo estimate of the true bagging estimate, approaching it as $B \to \infty$.

In Macroeconomics, Inoue and Kilian (2008) analyzed the usefulness of bagging in forecasting economic time series. The empirical results reveal that bagging can achieve expressive reductions in prediction mean-square errors in inflation forecasting, comparable to methods such as Ridge regression, LASSO, and Bayesian shrinkage predictor.

2.5 Machine Learning Models

Machine learning is a generic term that refers to techniques that allow computer systems to improve with experience and data. As defined by Goodfellow, Bengio and Courville (2016), *deep learning* is a type of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts. Hence, deep learning distinguishes itself by the versatility and predictive power that stem from the depth of the architecture of deep learning algorithms. It should be regarded as a subclass of machine learning, as shown in Figure 1.

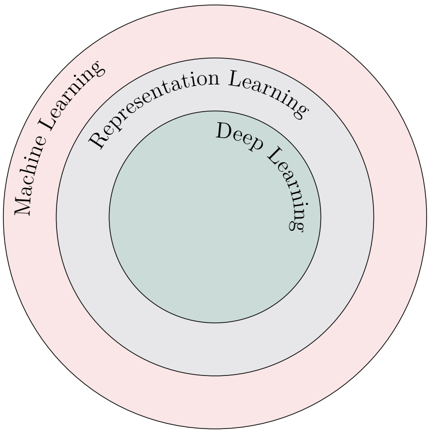


Figure 1 – Illustration of how machine learning, representation learning, and deep learning concepts are intertwined.

2.5.1 Random Forests

Random Forests is a model introduced and popularized by Breiman (2001). It involves the combination of tree predictors in a way such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (BREIMAN, 2001). In that sense, the model represents a substantial modification of bagging, since it builds a large collection of de-correlated trees for posterior averaging, as claimed by Hastie, Tibshirani and Friedman (2008). On many problems, the performance of Random Forests is comparable to boosting; however, they are simpler to train and tune (HASTIE; TIBSHIRANI; FRIEDMAN, 2008). Indeed, such properties emerge from the fact that Random Forests manage to reduce the variance of traditional regression trees, as advocated by Vasconcelos (2018).

In a broader view, Random Forests belong to the class of *regression trees* estimators. Examples are reproduced in Figure 2. Vasconcelos (2018) defines the method as a collection of flexible nonparametric models that recursively partition the set of independent variables into subsets, each modeled using regression methods. Through Figure 2, one may observe how a regression tree works. The bottom left panel unveils that, in this example, variables X_1 and X_2 were divided to produce five regions, R_j, $j = 1, \ldots, 5$, in which the dependent variable is predicted to be a constant c_j. Each region is defined according to the value assumed by X_1 and X_2. For instance, region R_1 is associated with $X_1 \leq t_1$ and $X_2 \leq t_2$, where t_k, $k = 1, 2, 3, 4$, are parameters to be estimated.

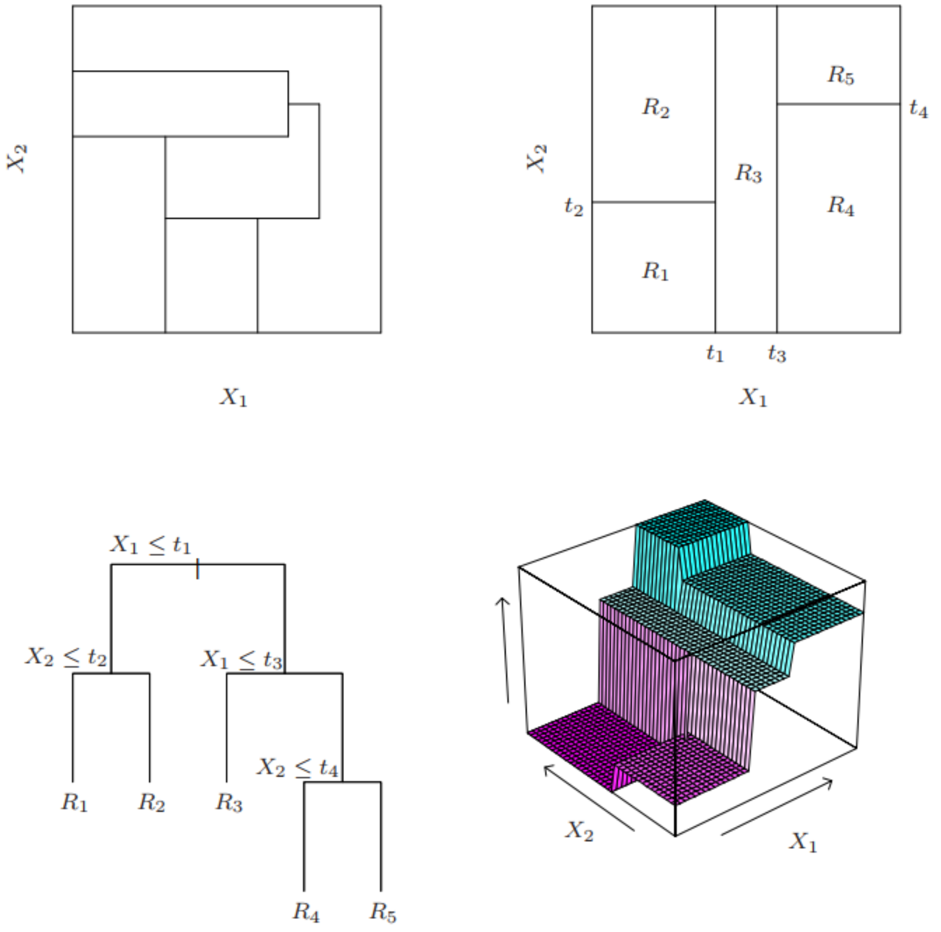


Figure 2 – Illustration of regression trees. The bottom right panel brings the perspective plot of the prediction surface associated with the tree appearing in the bottom left. Source: Hastie, Tibshirani and Friedman (2008).

According to Vasconcelos (2018), regression trees may be grown as follows. Assume that there are p explanatory variables such that $x_t = (x_{1,t}, \ldots, x_{p,t})$, for $t = 1, \ldots, T$, where $x_{i,t}$ is the observed value of the i-th variable X_i in period t. Proceeding backwards, suppose that, after determining splitting variables and points, M regions are found. Adopting as criterion the minimization of squared residuals, the constants $\hat{c}_m$, $m = 1, \ldots, M$, are defined as:

$$\hat{c}_m = \arg\min_{c_m} \sum_{t=1}^{T} I_{x_t \in R_m}(y_t - c_m)^2 = \frac{\sum_{t=1}^{T} I_{x_t \in R_m} y_t}{\sum_{t=1}^{T} I_{x_t \in R_m}} \tag{2.49}$$

where $I_{x_t \in R_m}$ is the indicator function that assumes value 1 if, and only if, $x_t \in R_m$, and value 0 elsewhere.

The growth of the regression tree is controlled through the sum of squared errors. Consider a splitting variable X_j and a split point s to partition the set of independent variables into two regions, namely, $R_1(j, s) = X|X_j \leq s$ and $R_2(j, s) = X|X_j > s$. The pair (j, s) is determined by solving:

$$\min_{j,s} \left[\min_{c_1} \sum_{t=1}^{T} I_{x_t \in R_1}(y_t - c_1)^2 + \min_{c_2} \sum_{t=1}^{T} I_{x_t \in R_2}(y_t - c_2)^2 \right] \tag{2.50}$$

The process is repeated iteratively on each of the resulting regions. Regarding the stopping criterion, one must initially recognize the trade-off between overfitting and underfitting related to the size of a tree. That is, a large tree may present several regions and parameters, providing a superb performance in-the-sample, but performing poorly out-of-sample. A small tree may be unable to capture the complexity of the data and also result in a lackluster out-of-sample accuracy.

An way to address this trade-off is the application of Random Forests. A Random Forest is a collection of trees, each specified in a bootstrapped sample of the original data (VASCONCELOS, 2018). The algorithm for regression or classification problems can be summarized as follows (HASTIE; TIBSHIRANI; FRIEDMAN, 2008):

1. For $b = 1$ to B:

   a) Draw a bootstrap sample $\boldsymbol{Z}$ of size N from the training data;

   b) Grow a random-forest tree T_B to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached: (i) select m variables at random from the p variables; (ii) pick the best variable/split-point among the m; and (iii) split the node into two daughter nodes.

2. Output the ensemble of trees $T_{b=1}^{B}$.

For regression problems, prediction at a new point x is carried out via:

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{2.51}$$

Likewise, in a classification setting, let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{rf}^{B}(x)$ is the majority vote of $\hat{C}_b(x)_{b=1}^{B}$. With respect to the parameter m, the default values are $m = p/3$ for regression and $m = \sqrt{p}$ for classification trees.

Breiman (2001) highlights the following desirable characteristics of Random Forests, explaining their widespread use in many practical applications:

1. Its accuracy is as strong as Adaboost and sometimes better;

2. There is robustness against outliers and noise;

3. It is faster than bagging or boosting;

4. It provides helpful internal estimates of error, strength, correlation, and variable importance;

5. It is simple and easily parallelized.

Due to its versatility and statistical power, several applications of Random Forests are found in the literature. For instance, Chen et al. (2017) propose a multiple Random Forests model, integrated with wavelet transforms, for the prediction of daily urban water consumption, achieving superior performance against the benchmarks. Meanwhile, Nti, Adekoya and Weyori (2019) develop a Random Forest model for the prediction of stock prices using macroeconomic variables as inputs, obtaining lower mean absolute errors compared with other time-series techniques.

2.5.2 Bayesian Regression Trees

The Bayesian Additive Regression Trees (BART) model was developed by Chipman, George and McCulloch (2010) as a "sum-of-tree" model where each regression tree is constrained by a regularization prior to be a weak learner. Fitting and inference can be accomplished through iterative Bayesian backfitting MCMC algorithm, generating samples from a posterior. As put by Chipman, George and McCulloch (2010), BART is effectively a nonparametric Bayesian regression approach which uses dimensionally adaptive random basis elements.

Mathematically, suppose that one desires to make inference about an function f that predicts an output Y using a p-dimensional vector of inputs $x = (x_1, \ldots, x_p)$ when:

$$Y = f(x) + \varepsilon, \quad \epsilon \sim N(0, \sigma^2) \tag{2.52}$$

It is feasible to model, or at least approximate, $f(x) = E(Y|x)$ by a sum of m regression tree such that:

$$f(x) \approx h(x) \equiv \sum_{j=1}^{m} g_j(x) \tag{2.53}$$

where each g_j denotes a regression tree. Thus, it is possible to approximate Equation 2.52 by a sum-of-trees model:

$$Y = h(x) + \varepsilon, \quad \epsilon \sim N(0, \sigma^2) \tag{2.54}$$

The essential idea here is to elaborate this model by imposing a prior that regularizes the fit by keeping the individual tree effects small (CHIPMAN; GEORGE; MCCULLOCH, 2010). As a consequence, the set formed by g_j becomes a dimensionally adaptive random basis of "weak learners" and, by weakening their effects, BART conceives a sum of trees, each of which explains a small and different portion of f.

Fitting is carried out via a tailored version of Bayesian back-fitting MCMC that iteratively constructs and fits successive residuals. Simultaneously, inference relies on iterations of the back-fitting algorithm, corresponding to MCMC samples from the induces posterior over the sum-of-trees model space (CHIPMAN; GEORGE; MCCULLOCH, 2010).

For time series prediction, an example of application is presented by Prüser (2019), who employs BART to forecast macroeconomic time series in a predictor-rich environment. The study allows to conclude that BART is a competitive model, exhibiting decent performance when handling high dimensional data sets in a macroeconomic context.

2.5.3 K-Nearest Neighbors

K-Nearest Neighbors (k-NN for short) is a popular nonparametric machine learning model. Although its foundations have been discovered back in the 1950s, Altman (1992) and others contributed to its extension and popularity. Essentially, it is a model used in classification and regression problems and trained using supervised learning. According to Hastie, Tibshirani and Friedman (2008), in the first case, given a query point x_0, k-NN finds k training points $x_{(r)}$, $r = 1, \ldots, k$ closest in distance to x_0, and then classify using majority vote among the k neighbors. There are multiple metrics to gauge distance. A popular one is the Euclidean distance, but other options exist, such as Mahalanobis distance.

Following Song et al. (2017), k-NN regression relies on learning by comparing given test instances with the training set. For a selected distance metric d, let $T = (x_1, y_1), \ldots, (x_n, y_n)$ be the training set. Also, let $x_i = (x_{i1}, \ldots, x_{im})$ be the ith instance denoted by m attributes and associated with output y_i, and n be the number of instances. For a given test instance x, for which a prediction $\hat{y}$ is desired, the algorithm proceed as follows. First, the points x_i are ranked in terms of the distance d_i to x. Considering the k points closest to x, the forecast $\hat{y}$ is defined as:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i(x) \tag{2.55}$$

where $y_i(x)$ depends on x because the k points selected are based on their distance to x.

2.5.4 Support Vector Regression

Support Vector Regression (SVR) is a method that modifies the standard OLS so as to control how much error is acceptable in the fitted model. For that purpose, SVR minimizes the coefficients or, in fact, the ℓ_2-norm of the coefficient vector, while constraining the absolute error to be less than or equal a predefined margin. SVR is inspired in Support Vector Classifiers (SVC) and inherits some of its properties.

As detailed in Hastie, Tibshirani and Friedman (2008), first, assume a generic linear regression model:

$$f(x) = x^t\beta + \beta_0 \tag{2.56}$$

The parameter β is estimated by minimizing:

$$H\left(\beta, \beta_0\right) = \sum_{i=1}^{N} V\left(y_i - f\left(x_i\right)\right) + \frac{\lambda}{2}\|\beta\|^2 \tag{2.57}$$

where:

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon \\ |r| - \epsilon, & \text{otherwise} \end{cases} \tag{2.58}$$

is an ϵ-insensitive error measure, ignoring errors of size less than ϵ.

If $\hat{\beta}$ and $\hat{\beta}_0$ are the minimizers of H, the solution function can be show to have the form (HASTIE; TIBSHIRANI; FRIEDMAN, 2008):

$$\hat{\beta} = \sum_{i=1}^{N} \left(\hat{\alpha}_i^* - \hat{\alpha}_i\right) x_i \tag{2.59}$$

$$\hat{f}(x) = \sum_{i=1}^{N} \left(\hat{\alpha}_i^* - \hat{\alpha}_i\right) \langle x, x_i \rangle + \beta_0 \tag{2.60}$$

where $\langle \cdot, \cdot \rangle$ is the canonical inner product and $\hat{\alpha}_i$ and $\hat{\alpha}_i^*$ are positive and solve the quadratic programming problem:

$$\min_{\alpha_i, \alpha_i^*} \epsilon \sum_{i=1}^{N} \left(\alpha_i^* + \alpha_i\right) - \sum_{i=1}^{N} y_i \left(\alpha_i^* - \alpha_i\right) + \frac{1}{2} \sum_{i,i'=1}^{N} \left(\alpha_i^* - \alpha_i\right)\left(\alpha_{i'}^* - \alpha_{i'}\right) \langle x_i, x_{i'} \rangle \tag{2.61}$$

subject to the constraints:

$$\begin{gathered} 0 \leq \alpha_i, \alpha_i^* \leq 1/\lambda \\ \textstyle\sum_{i=1}^{N} \left(\alpha_i^* - \alpha_i\right) = 0 \\ \alpha_i \alpha_i^* = 0 \end{gathered} \tag{2.62}$$

Due to the nature of these constraints, generally only a subset of the solutions $\left(\hat{\alpha}_i^* - \hat{\alpha}_i\right)$ are nonzero, and the associated data values are called the *support vectors*.

The regression analysis can also be written in terms of a set of basis functions $\{h_m(x)\}$, $m = 1, 2, \ldots, M$:

$$f(x) = \sum_{m=1}^{M} \beta_m h_m(x) + \beta_0 \tag{2.63}$$

To estimate β and β_0, we minimize:

$$H\left(\beta, \beta_0\right) = \sum_{i=1}^{N} V\left(y_i - f\left(x_i\right)\right) + \frac{\lambda}{2} \sum \beta_m^2 \tag{2.64}$$

for some general error function V. The solution has the form:

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{a}_i K\left(x, x_i\right) \tag{2.65}$$

with:

$$K(x, y) = \sum_{m=1}^{M} h_m(x) h_m(y) \tag{2.66}$$

2.6 Deep Learning

Deep learning is built around the hypothesis that a deep, highly hierarchical model can be more efficient at representing some functions than a shallow one. Therefore, deep learning encompasses several neural network architectures formed by multiple hidden layers, in opposition to shallow networks. Until 2006, concerns regarding the obstacles to train deep networks explained their lack of popularity despite their potential. Indeed, Schmidhuber (2015) recollects that, back in the 1990s, deep feedforward or recurrent networks were hard to train by backpropagation, the most popular learning algorithm until then, the major reason being vanishing or exploding gradients during optimization, which became known as the *Fundamental Deep Learning Problem*.

A breakthrough occurred promoted by the seminal paper by Hinton, Osindero and Teh (2006), who showed that deep belief networks could be efficiently trained using a strategy called greedy layer-wise pretraining. Over the years, additional alternatives to overcome the Fundamental Deep Learning Problem were introduced, such as hessian-free optimization algorithms and GPU-based computers, which have million times the computation power of CPUs of the early 1990s (SCHMIDHUBER, 2015). Nowadays, these advances ensured the dissemination of deep learning models in many areas of research and in several practical applications.

Corroborating the idea that deep learning yields more powerful models due to the existence of deep hierarchies, Eldan and Shamir (2016) mathematically demonstrate that increasing the depth of neural networks is generally more effective than increasing the number of neurons in the existing layers in terms of out-of-sample performance. As illustration, they show that some classes of functions in $\mathbb{R}^d$, expressible by a small 3-layer feedforward neural network, cannot be approximated by any 2-layer network to more than a predefined accuracy unless the number of neurons per layer grows exponentially in the dimension d. The result is independent of the activation function.

The superiority of deep learning model over shallow networks is also supported by other theoretical findings recently published. For instance, Delalleau and Bengio (2011) present families of functions that can be represented much more efficiently (i.e with fewer hidden units) via deep networks instead of shallow ones. Furthermore, Roux and Bengio (2010) demonstrate that deep, but narrow generative networks do not require more parameters than shallow ones to achieve universal approximation.

2.6.1 Artificial Neural Networks

Prior to introducing advanced deep learning models, it is necessary to begin with artificial neural networks, their cornerstone. *Artificial neural networks* (ANN), or simply *neural networks* (NN), have become a widespread and highly flexible machine learning model. According to Haykin (2004), a neural network is a massively parallel distributed processor made up of simple processing units (*neurons*), which has a natural propensity for storing experiential knowledge and making it available for use. In that sense, they resemble the brain in two aspects:

1. Knowledge is obtained by the networks from its environment through a *learning process*; and

2. Interneuron connection strengths, known as *synaptic weights* (or simply *weights*), are used to store the acquired knowledge.

The aforementioned learning process, through which the network is capable of extracting and extrapolating knowledge from observations, is called a *learning algorithm*. This process is responsible for iteratively adjusting the synaptic weights so as to attain a certain objective, such as minimizing a loss function that measures deviations of forecast values from observed ones.

There is a wide availability of algorithms for this purpose, and they are broadly classified in two groups: *supervised* and *unsupervised*. The first is comprised by algorithms that require a dataset formed by mapped inputs and outputs such that the synaptic weights are adjusted in order to minimize some loss function. The second group contains algorithms that do not demand labeled outputs when optimizing synaptic weights. Other variants of the learning paradigm are possible; for instance, *semi-supervised learning* uses some examples including supervision targets, while the remaining do not (GOODFELLOW; BENGIO; COURVILLE, 2016).

As emphasized by Haykin (2004), neural networks derive their computing power via their highly parallel distributed structure and, seconds, their ability to learn and, hence, generalize. *Generalization* relates to out-of-sample performance, in the sense that these models present enough flexibility and power to yield decent predictions for inputs not encountered during the learning (also known as *training*) phase. Naturally, when training any neural network, the main objective must be to achieve a solid generalization power, implying a low out-of-sample loss. Focusing solely on minimizing the in-sample loss may induce overfitting.

The basic neuron model described above can be represented as a series of functional transformations, according to Bishop (2006). Figure 3 displays a typical neuron. For simplicity, assume that this unit receives n inputs given by $x_1, x_2, \ldots, x_n$. Mathematically, the operation carried out by the neuron, which is denoted *propagation*, can be summarized as follows. Primarily, a linear combination z of the inputs is computed using the corresponding weights w_i to each input x_i. Thus:

$$z = w_0 + \sum_{i=1}^{n} w_i x_i \tag{2.67}$$

where w_0 is a *bias* introduced for additional versatility (i.e. generalization power). This linear combination is also known as *activation*. In a given network, the activation of each neuron is inputted in the respective *activation function* f. The output of a neuron then becomes:

$$y = f(z) \tag{2.68}$$

This output is then transmitted forward to the neurons to which the neuron in question is connected. The process is repeated until the final (output) layer of the network.

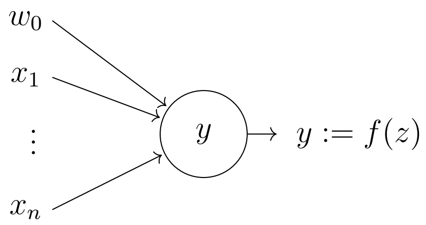


Figure 3 – Single processing unit and its components. The activation function is denoted by f and applied on the actual input z of the unit to form its output $y = f(z)$. $x_1, \ldots, x_n$ represent input from other units within the network; w_0 is called bias and represents an external input to the unit. All inputs are mapped onto the actual input z using the propagation rule.

With respect to the activation function f, many choices are available in the literature. Typically, the following functions are used with this purpose:

1. Threshold (Heaviside): it corresponds to a binary function, implying that, if the input value is above a certain threshold (commonly 0), the neuron is activated and sends a signal to the next layer independent of the input value:

$$f(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \tag{2.69}$$

2. Linear: generates a signal proportional to the input. It suffers from two limitations. First, since its derivative is always a constant, learning via backpropagation is not feasible. Second, a neural network formed by layers of linear units is essentially a regression model, for successive linear transformations across the layers is intrinsically equivalent to a single linear layer, also establishing a linear relationship between input and output. The function is given by:

$$f(x) = ax \tag{2.70}$$

where a is a parameter.

3. Sigmoid (or logistic): the popularity of this activation function is explained by several factors. First, it is differentiable in every point of its domain. Second, the function is monotonic and its gradient is smooth, preventing jumps in output values. Also, outputs lie between 0 and 1 (see Equation 2.71), imposing a normalization that is beneficial when predicting probabilities, for instance. However, there are disadvantages, such as vanishing gradients for large $|x|$. The sigmoid function is defined as:
$$f(x) = \frac{1}{1 + e^{-\lambda x}} \tag{2.71}$$
where λ is a parameter.

4. Hyperbolic tangent: it is an adaptation of the sigmoid function that produces zero-centered outputs in the range $[-1, 1]$. Additionally, its gradient is more steep, suffering less intensively from the vanishing gradient problem. The hyperbolic tangent can be written as:
$$f(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.72}$$

However, in many contexts, there is no consensus regarding which activation function performs better in terms of test error. As illustration, for stock market forecasting, Atsalakis and Valavanis (2009) survey more than 100 related published articles that focus on neural and neuro-fuzzy models for this purpose and conclude that there is no unanimity concerning the best network architecture for this problem. Meanwhile, Sibi, Jones and Siddarth (2013) demonstrate that the activation function chosen has a statistically significant impact on the model performance.

In 2011, the seminal paper by Glorot, Bordes and Bengio (2011) proposed a new activation function called *rectifier linear unit*, or *ReLU*, defined as:

$$\mathit{ReLU}(x) = \max(0, x) \tag{2.73}$$

This function was created by observing that, while logistic sigmoid neurons are more biologically plausible than hyperbolic tangent neurons, the latter works better for training multi-layer networks. In this aspect, rectifying neurons are an even better model of biological neurons and yield equal or better performance than previous functions.

The rectifier activation function allows a network to effortlessly achieve sparse representations due to the hard nonlinearity and nondifferentiability at zero. Even though a hard saturation is imposed at this point, which may compromise optimization by blocking gradient back-propagation, the experimental results offered by Glorot, Bordes and Bengio (2011) suggest that this condition may actually help supervised training. The authors hypothesize that the hard non-linearities do not threaten the model stability as long as the gradient can propagate along some paths, i.e., that some of the hidden units in each layer are non-zero.

One disadvantage of the ReLU is that, due to the abrupt activation at 0, units are deactivated when the input is negative, slowing the learning process. A solution is presented by Maas, Hannun and Ng (2013), who introduced the *Leaky ReLU*,or LReLU, which can be written as:

$$LReLU(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.01x, & \text{if } x \leq 0 \end{cases} \tag{2.74}$$

As it is possible to infer, LReLU allows for a small, non-zero gradient when the unit is saturated and not active (MAAS; HANNUN; NG, 2013). The factor 0.01 may be adjusted as required.

Another alternative has been recently introduced by Klambauer et al. (2017). The authors developed *self-normalizing networks* (SNN) to enable high-level abstract representations. In this sense, while batch normalization requires explicit normalization, neuron activations in a SNN naturally converge towards zero mean and unit variance, even under the presence of noise and perturbations. In particular, neurons possess a novel activation function named *scaled exponential linear units* (SELU), which induces self-normalizing properties. The SELU function is defined by:

$$SELU(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \leq 0 \end{cases} \tag{2.75}$$

where $\lambda > 1$ to ensure a slope greater than one for positive net inputs. Interesting properties emerge from this formulation. Indeed, it may be shown that SELU fulfills the following requirements for an activation function (KLAMBAUER et al., 2017):

1. Negative and positive values for controlling the mean;
2. Saturation regions (i.e. intervals where the derivative approaches zero) to dampen the variance whenever it becomes large in the lower layer;
3. A slope greater than one to increase the variance whenever it becomes too small in the lower layer; and
4. A continuous curve in its domain, ensuring the existence of fixed points.

2.6.2 Deep Multilayer Perceptron

The *multilayer perceptron* (MLP) is the most basic type of ANN. In its most vanilla form, it is formed by an input layer, a hidden layer, and an output layer. *Deep multilayer perceptrons*, depicted in Figure 4, present multiple hidden layers in sequence, adding more flexibility and complexity to the network in exchange of greater generalization power. By definition, these are *feedforward networks*, in the sense that they are acyclic, i.e. data flows from the input layer toward the output layer without any kind of recurrence.

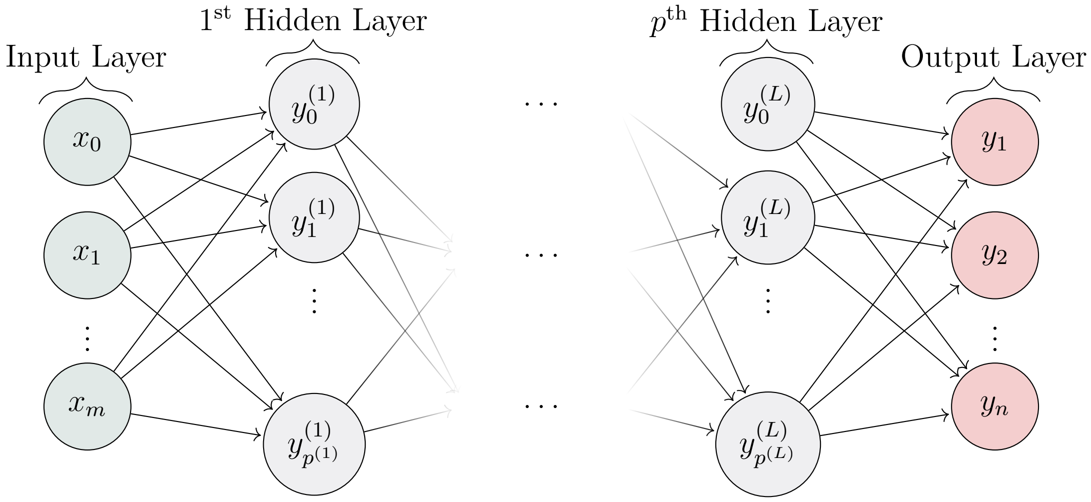


Figure 4 – Network graph of a $(L+1)$-layer deep perceptron with m input units and n output units. The l^{th} hidden layer contains $p^{(l)}$ hidden units.

As demonstrated by Hornik, Stinchcombe and White (1989), under rather general conditions, deep feedforward networks can universally approximate any measurable function with a predefined accuracy. This result is called *Universal Approximation Theorem* (HAYKIN, 2004), and analogous versions are available for other kinds of networks. Therefore, MLPs are powerful models for a wide collection of prediction problems.

MLPs and other networks are commonly trained via *backpropagation*, one of the most important learning algorithms in the field, since it is effective on its own and it is the basis for more advanced ones. Essentially, it can be seen as a generalization of the OLS. Indeed, assume that, in the training sample, the error of the output node j in the n-th data point is:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.76}$$

where $d_j(n)$ is the observed value and $y_j(n)$ is the target value. In addition, it is assumed a quadratic loss function $L(n)$ of the form:

$$L(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{2.77}$$

By gradient descent, the change in each weight so as to minimize L is:

$$\Delta w_{ji}(n) = -\eta \frac{\partial L(n)}{\partial z_j(n)} y_i(n) \tag{2.78}$$

where y_i is the output of the previous neuron i of the preceding layer and η is the predefined *learning rate.*

The derivative of the previous equation depends on the activation $z_j(n)$. It is possible to prove that:

$$\frac{\partial L(n)}{\partial z_j(n)} = e_j(n) f'(z_j(n)) \tag{2.79}$$

where f is the activation function and f', its derivative. For hidden nodes, the result above becomes:

$$\frac{\partial L(n)}{\partial z_j(n)} = -f'(z_j(n)) \sum_k \frac{\partial L(n)}{\partial z_k(n)} w_{kj}(n) \tag{2.80}$$

Since it depends on gradient descent, there is no guarantee that backpropagation will be able to find the global minimum of the error function. Indeed, the loss functions of neural networks are highly non-convex due to the intrinsic nonlinearities of such models, generating functions with multiple local minima.

2.6.3 Deep Autoencoder

An *autoencoder* is a neural network that is trained to attempt to copy its input to its output (GOODFELLOW; BENGIO; COURVILLE, 2016). According to Vincent et al. (2010), the concept behind autoencoders can be evolved from the idea of transforming and retaining information about the inputs of a neural network. More formally, we are interested in learning a possibly stochastic mapping from input X to a novel representation Y. Mathematically, one wants to find the vector of parameters θ that defines the mapping:

$$q(Y|X;\theta) = q(Y|X) \tag{2.81}$$

Naturally, a reasonable criterion that any good mapping must meet, at least to some degree, is to retain a significant amount of information about the input. As formulated by Vincent et al. (2010), this problem can be expressed in information-theoretic terms as maximizing the mutual information $I(X,Y)$ between an input random variable X and its higher level of representation Y. This is called *infomax principle.* This mutual information can be decomposed as:

$$I(X,Y) = H(X) - H(X|Y) \tag{2.82}$$

where H is the information entropy. Since the observed input X comes from an unknown distribution $q(X)$ on which θ has no influence, the infomax principle for this function can be simply written as:

$$\arg\max_\theta I(X,Y) = \arg\max_\theta -H(X|Y) = \arg\max_\theta E_{q(X,Y)}[\ln q(X|Y)] \tag{2.83}$$

Now, for any distribution $p(X|Y)$, we have (VINCENT et al., 2010):

$$E_{q(X,Y)}[\ln p(X \mid Y)] \leq \underbrace{E_{q(X,Y)}[\ln q(X \mid Y)]}_{-H(X|Y)} \tag{2.84}$$

Moving on, considering a parametric distribution $p(X|Y;\theta')$, parameterized by θ', and the following optimization:

$$\max_{\theta,\theta'} E_{q(X,Y;\theta)} \left[\ln p \left(X \mid Y;\theta'\right)\right] \tag{2.85}$$

According to Equation 2.84, this problem is equivalent to maximizing a lower bound on $-H(X|Y)$ and, thus, on the mutual information. Furthermore, if an additional constraint is incorporated so as to restrict ourselves to a deterministic mapping from X to Y, meaning that a representation Y is to be computed by a function $Y = f_\theta(X)$ parameterized on θ, which corresponds to setting:

$$q(Y|X;\theta) = \delta\left(Y - f_\theta(X)\right) \tag{2.86}$$

where δ denotes the Dirac's delta function, then the optimization can be expressed as:

$$\max_{\theta,\theta'} E_{q(X)}\left[\ln p\left(X \mid Y = f_\theta(X); \theta'\right)\right] \tag{2.87}$$

once again corresponding to maximizing a lower bound on the mutual information (VINCENT et al., 2010).

With the development presented so far, it is possible to build the *reconstruction error* criterion used to train autoencoders. Indeed, since $q(X)$ is unknown, but may be sampled from it, the empirical average over the training samples can be used instead as an unbiased estimated, which implies replacing $E_{q(X)}$ by $E_{q^0(X)}$:

$$\max_{\theta,\theta'} \mathbb{E}_{q^0(X)}\left[\ln p\left(X \mid Y = f_\theta(X); \theta'\right)\right] \tag{2.88}$$

Building on this development, training an autoencoder to minimize reconstruction error amounts to maximizing a lower bound on the mutual information between input X and learnt representation Y (VINCENT et al., 2010). The traditional autoencoder is depicted on Figure 5, where it is possible to see that the network has two elements: (1) the *encoder*, which compresses the input and reduces its dimension; and (2) the *decoder*, which is responsible for decoding the compressed data and generate outputs as close as possible to the input data.

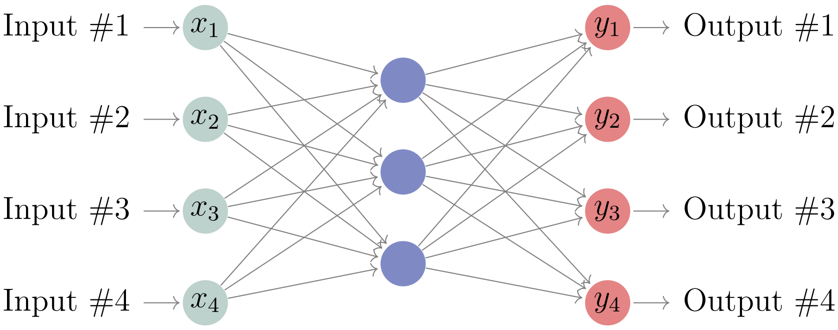


Figure 5 – Representation of an autoencoder with a single hidden layer. The blue circles denote the nodes where the encoding process occurs. The red nodes decompress the encoded data, generating outputs approximating the inputs inserted in the green nodes.

Using the prior formulas, the encoder is a deterministic mapping f_θ that transforms an input vector $\mathbf{x}$ into a hidden representation:

$$\mathbf{y} = f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.89}$$

with parameter set $\theta = \{\mathbf{W}, \mathbf{b}\}$ and activation function s.

The resulting hidden representation is then converted back into a reconstructed vector $\mathbf{z}$ in the input space, $\mathbf{z} = g_{\theta'}(\mathbf{y})$. This mapping is called the decoder, which is mathematically expressed as:

$$g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}') \tag{2.90}$$

with parameter set $\theta' = \{\mathbf{W}', \mathbf{b}'\}$ and activation function s.

As argued by Vincent et al. (2010), $\mathbf{z}$ is not to be regarded as an exact reproduction of $\mathbf{x}$, but rather in probabilistic terms as the parameters (typically the mean) of a distribution $p(X|Z = \mathbf{z})$ that may generate with high likelihood. Therefore, the reconstruction error to be optimized is:

$$L(\mathbf{x}, \mathbf{z}) \propto -\ln p(\mathbf{x}|\mathbf{z}) \tag{2.91}$$

Common choices for p include the normal distribution, when dealing with continuous variables, yielding a quadratic loss function, and the Bernoulli distribution for binary variables, producing the cross-entropy loss.

Some common extensions of autoencoders include:

1. *Sparse autoencoders*, which simply modifies the training criterion by adding a sparsity penalty $\Omega(\boldsymbol{h})$ on the code layer $\boldsymbol{h}$, in addition to the reconstruction error (GOODFELLOW; BENGIO; COURVILLE, 2016). Therefore, the optimization criterion is based on the minimization of a loss function of the form:

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h}) \tag{2.92}$$

2. *Denoising autoencoders*: as detailed by Goodfellow, Bengio and Courville (2016), rather than adding a penalty Ω to the cost function, a denoising autoencoder learns by minimizing a loss function that involves a corrupted copy $\tilde{\boldsymbol{x}}$ of $\boldsymbol{x}$ that has been altered by some form of noise. Therefore, a denoising autoencoder must undo this corruption rather than simply copying their input. The advantage, as contended and shown by Lu et al. (2017), is that, along with the ability of handling noise, the autoencoder gains greater capacity of learning properties of the generative process responsible for the input data.

Applications of autoencoders in real-world problems are varied. These networks have been successfully trained to implement dimensionality reduction and information retrieval, which is the task of finding entries in a database that resemble a query entry (GOODFELLOW; BENGIO; COURVILLE, 2016). In particular, lower-dimensional representation can enhance performance on many tasks, and Bengio, Courville and Vincent (2013) assert that the machine learning algorithms are heavily influenced by the data representation method.

According to Wang, Yao and Zhao (2016), the use of autoencoders in experiments with real and simulated data corroborates their dimension reduction ability for datasets

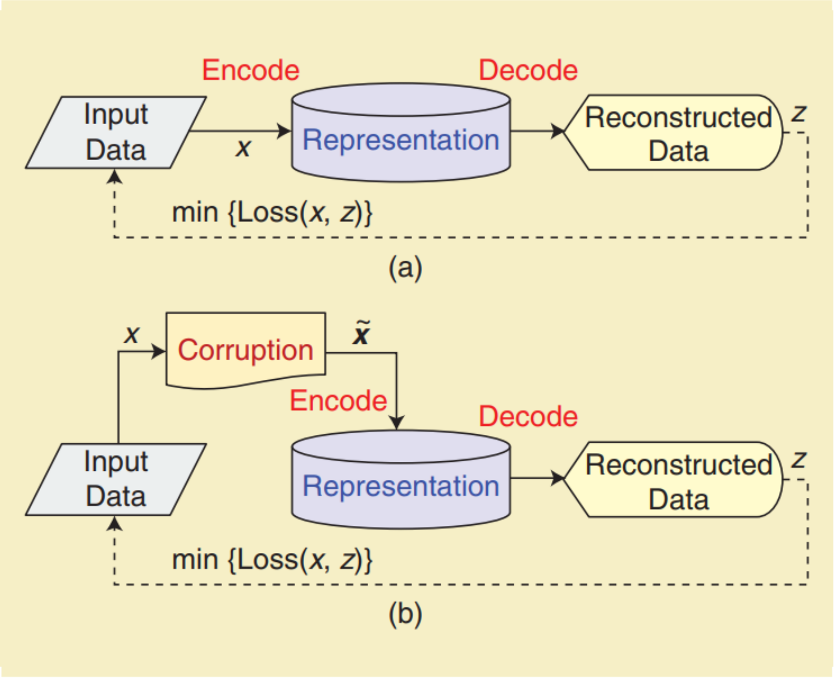


Figure 6 – Prototypes of (a) a conventional autoencoder and (b) a denoising autoencoder. Source: Dong et al. (2018).

comprised by time series. Furthermore, the authors observe not only the effectiveness of autoencoders in this task, but also its potential to unveil nontrivial patterns in the data. In this aspect, they outperform traditional techniques, such as PCA, LDA (Linear Discriminant Analysis), LLE (Locally Linear Embedding) and Isomap (WANG; YAO; ZHAO, 2016).

2.6.4 Convolutional Neural Networks

Convolutional Neural Networks, or *CNNs*, are a specialized kind of neural network for processing data that has a known, grid-like topology (GOODFELLOW; BENGIO; COURVILLE, 2016). Examples include time series, which can be interpreted as a 1D grid taking samples at regular time intervals, and image data, which can be regarded as a 2D grid of pixels. The name given to these networks is explained by the use of convolution operations in some of their layers.

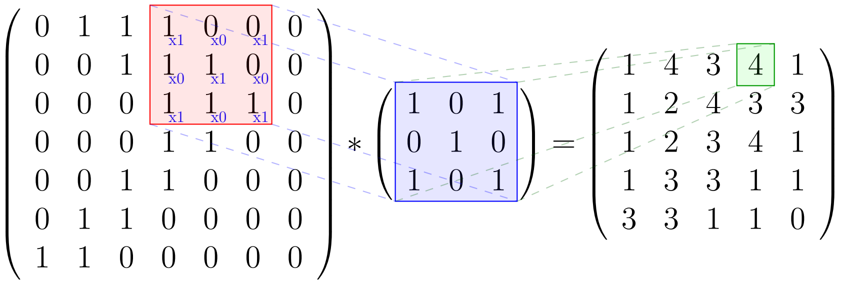
Mathematically, the *convolution* of two functions f and g over an infinite range is:

$$s(t) = (x * w)(t) \equiv \int_{-\infty}^{\infty} x(v)w(t-v)dv \tag{2.93}$$

assuming the integral of the previous equation is defined. Following the terminology commonly referred to, the first argument, x, is denominated *input*, while the second argument, w, is known as *kernel* (GOODFELLOW; BENGIO; COURVILLE, 2016). The output, s, is sometimes called *feature map*. In discrete time, for univariate functions, the convolution is defined as:

$$s(t) = (x * w)(t) = \sum_{v=-\infty}^{\infty} x(v)w(t-v) \tag{2.94}$$

This expression can be readily extended for multivariate functions. In practice, for machine learning applications, the input is typically a tensor of data, while the kernel is a multidimensional array of parameters optimized by the learning algorithm. The convolution transformation for matrices is outlined in Figure 7, while the mapping is shown in Figure 8.



$$\begin{pmatrix} 0 & 1 & 1 & 1_{\times 1} & 0_{\times 0} & 0_{\times 1} & 0 \\ 0 & 0 & 1 & 1_{\times 0} & 1_{\times 1} & 0_{\times 0} & 0 \\ 0 & 0 & 0 & 1_{\times 1} & 1_{\times 0} & 1_{\times 1} & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & \boxed{4} & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix}$$

Figure 7 – Example of convolution between two matrices, as frequently carried out by machine learning models.

Even though common matrix multiplication is replaced by convolutions in CNNs, these models also belong to the class of feedforward networks, for, as emphasized by Rawat and Wang (2017), the information flow tales place in a single direction only, from the inputs to the outputs of the network. The architecture underpinning CNNs is motivated by the visual cortex in the brain, which consists of alternating layers of simple and complex cells (RAWAT; WANG, 2017). Therefore, concerning data flow, CNNs operate like a fully connected neural network such as MLP.

Goodfellow, Bengio and Courville (2016) highlights that CNNs leverage three important features that can help improve a machine learning system: sparse interactions

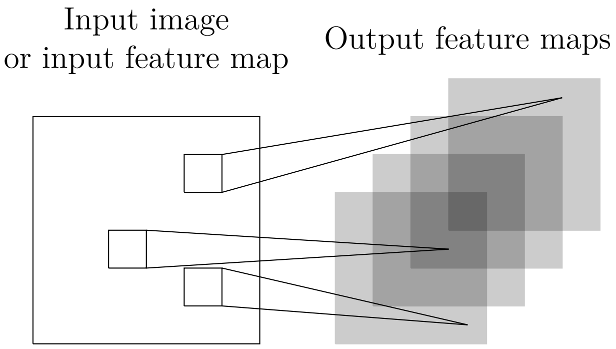


Figure 8 – Illustration of a single convolutional layer. If layer l is a convolutional layer, the input image (if $l = 1$) or a feature map of the previous layer is convolved by different filters to yield the output feature maps of layer l.

(or sparse weights), parameter sharing, and equivalent representations. The first feature of CNNs, the sparse interactions, are accomplished by making the kernel smaller than the input. The implication is that fewer parameters have to be stored, reducing memory requirements, improving statistical efficiency, and computing the output with less mathematical operations.

Parameter sharing is related to the use of kernels to process the inputs. It refers to using the same parameter for more than one function in the model. In that case, the kernel is the same for every position of the inputs. Hence, instead of learning a set of parameters for every location, a single set is learned and applied to every input. Not only this design induces computational efficiency, but it also leads to statistical gains and is more consistent with the nature of some problems tackled with neural networks.

Furthermore, parameter sharing creates another interesting feature, equivariance to translation. Formally, a function $f(x)$ is *equivariant* to a function $g(x)$ if $f(g(x)) = g(f(x))$. Convolutions are equivariant to any function that translates (i.e. shifts) the input. As shown by Goodfellow, Bengio and Courville (2016), for inputs represented by time series, if we move an event later in time in the input, the exact same representation of it will appear in the input, just shifted in time. With respect to other possible transformations, such as rotation or scale changes, convolution is not naturally equivariant, requiring specific architectures for handling these types of transformations.

Besides the existence of convolutional layers, pooling (subsampling) layers are also frequent. Indeed, standard CNN generally consists of two types of layers: the convolutional layer, in which the convolution operation is applied, and the pooling layer, where the number of parameters and the spatial size of the representation are reduced (SEZER; OZBAYOGLU, 2018). Typically, in the last subsampling layer, the data becomes a one-dimensional vector and is inputted to fully connected networks such as MLPs. Dhillon and Verma (2020) offers a more general view, identifying convolutional, pooling, fully connected, and normalization layers in an extended CNN. Figure 9 portrays a typical convolutional network. A review of modern CNN architectures is provided by Dhillon and Verma (2020).

Although many architectures are available in the literature, Goodfellow, Bengio and Courville (2016) highlight that a CNN typically carries out three set of operations:

1. First stage: the layer receives the input and performs several convolutions in parallel to produce a set of linear activations. CNNs are flexible enough to work with inputs of variable size;

2. Second stage (or detector stage): each linear activation is run through a nonlinear activation function;

3. Third stage: a pooling function is used to modify the output of the layer so as to make it usable for the next layer.

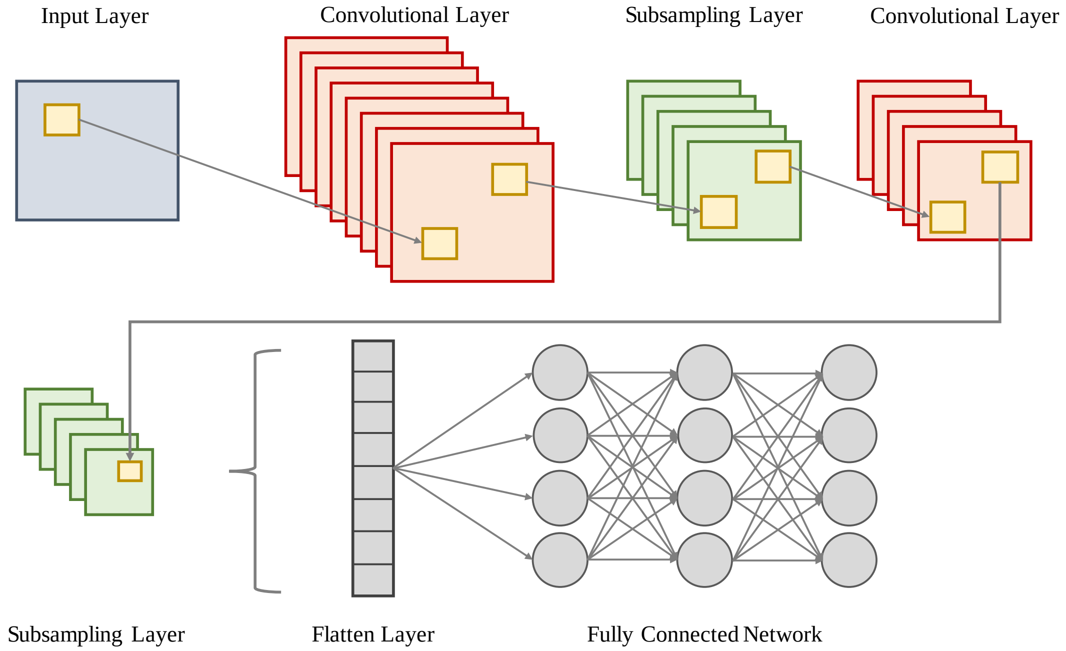


Figure 9 – General representation of a convolutional neural network. The yellow rectangle represents the kernel.

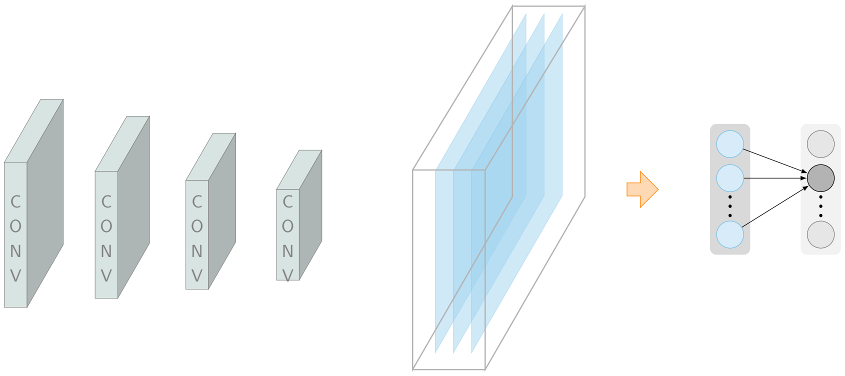


Figure 10 – Stacked CNNs and representation of the *pooling* function.

CNNs have been successfully applied to image processing; see Bai, Tang and An (2019) and Yamashita et al. (2018). Visual object recognition has also seen interesting applications involving CNNs, as shown by LeCun et al. (2010). Meanwhile, Zhao et al. (2017) develop a time-series classification system based on CNNs.

2.6.5 Recurring Neural Networks

Arising from the ideas developed by Rumelhart, Hinton and Williams (1986), Recurrent Neural Networks (RNN) form another set of promising deep learning algorithms in terms of their ability to cope with temporal dependency in data. As claimed by Hüsken and Stagge (2003), RNNs are dynamical systems that efficiently use the temporal information in the input sequence, both for classification as well as for prediction. This feature is useful in many contexts, from text interpretation to time series modeling. Its prominence is also explained by the *Universal Approximation Theorem*, since it is also possible to show that RNN share a similar property with deep feedforward networks: they are able to approximate almost any recurrent system of equations, under very general conditions (GOODFELLOW; BENGIO; COURVILLE, 2016; SCHÄFER; ZIMMERMAN, 2007).

Briefly, RNNs are a family of neural networks for processing sequential data, extending traditional feedforward networks by preserving their temporal dimension (GOODFELLOW; BENGIO; COURVILLE, 2016; ESSIEN; GIANNETTI, 2019). In terms of architecture, RNNs are models that capture temporal dependencies using cycles in their computational graph, as shown in Figure 11. One of the main advantages of this strategy is that parameters are shared across multiple parts of the model. As observed by Goodfellow, Bengio and Courville (2016), this behavior increases the generalization ability and makes it possible to employ the same model for different types of data, with distinct formats (lengths, for instance).

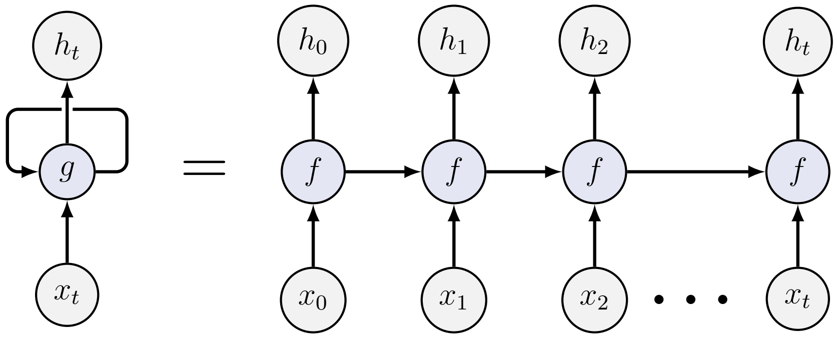


Figure 11 – Example of a RNN with no output. The RNN simply receives the input x_t and incorporates into the state h_t that is passed forward through time. On the right, the same network seen as an unfolded computational graph (GOODFELLOW; BENGIO; COURVILLE, 2016).

Essentially, in Figure 11, the mathematical operation carried out by the RNN can be summarized as:

$$h_t = g(x_t, x_{t-1}, \ldots, x_2, x_1) = f(h_{t-1}, x_t) \tag{2.95}$$

where h_t is the state passed forward through time, x_t are the temporal inputs, and f and g are equivalent functional representations of the recurrence implemented by the RNN.

Applications of RNN are offered, for instance, by Zhang, Wu and Chang (2018), who employ these networks in the domain of short-term load forecasting. Besides, Hsieh, Hsiao and Yeh (2011) demonstrate that RNNs may be coupled with wavelets for stock market forecasting.

2.6.6 Long Short-Term Memory (LSTM) Networks

Even though RNNs are promising for time series modeling, it is frequently observed that such networks are daunting to train and unable to detect and reproduce long-term autocorrelations. This drawback is explained by the vanishing gradient problem (PALANGI; WARD; DENG, 2016; SHEN et al., 2018), arising when neural networks are trained using gradient-based learning methods. Such condition prevents weights from changing and adapting in face of serially correlated data with polynomial decay, decreasing their statistical power. Even though this limitation may be attenuated using more flexible activation functions, such as ReLU (GLOROT; BORDES; BENGIO, 2011), more efficient solutions must be implemented to fully address this issue.

As stated by Schmidhuber (2015), experiments suggest that vanishing or exploding gradients are common both in deep feedforward networks and RNNs. Indeed, with cumulative backpropagated error, gradients either shrink rapidly or grow exponentially. In a RNN, this issue seems more pervasive due to the recurrences appearing in the network. As an example, assuming a simple recurrence of the form:

$$\mathbf{x}_n = \mathbf{W}^t \mathbf{x}_{n-1} \tag{2.96}$$

The general term may be written as a function of the initial value $\mathbf{x}_0$ as:

$$\mathbf{x}_n = (\mathbf{W}^t)^n \mathbf{x}_0 \tag{2.97}$$

and if $\mathbf{W}$ admits an eigendecomposition of the form:

$$\mathbf{W} = \mathbf{A}\mathbf{D}\mathbf{A}^t \tag{2.98}$$

with orthogonal $\mathbf{A}$, then the recurrence becomes:

$$\mathbf{x}_n = \mathbf{A}^t (\mathbf{D}^t)^n \mathbf{A}\mathbf{x}_0 \tag{2.99}$$

In the expression above, as $n \to +\infty$, eigenvalues with absolute value above 1 explode, whereas the remaining decay exponentially to zero. This simplified dynamic system is useful to explain the origin of the vanishing and exploding gradients in RNNs.

With the purpose of overcome the limitations of traditional RNNs, different architectures have been advanced in the literature. For instance, according to Goodfellow, Bengio and Courville (2016), one of the most effective sequence models used in practical applications are called *gated RNNs*. These models are based on the idea of creating paths through time that have derivatives that neither vanish nor explode. The connection weights between gated RNNs are adjusted at each time step.

The main advantage of these networks is their ability to accumulate information over a long duration. In many fields, this property is desirable. For instance, in text processing, the meaning of a word depends on the position in which it occurs in the sentence, and on previous and upcoming words. Time series modeling presents the same challenges, since forecasting requires processing and extracting features of past observations due to the existence of autocorrelations.

As an illustration, Shen et al. (2018) provide an application of gated recurrent unit (GRU) networks for predicting trading signals for stock indices. Experiments show that these models outperform SVMs and other benchmarks in terms of prediction accuracy. At the same time, Niu et al. (2020) develop a modified GRU network with a feature selection mechanism for wind power forecasting, obtaining superior results in comparison to traditional benchmarks.

In that context, Long Short-Term Memory (LSTM) networks are another alternative. They are a particularization of RNNs, distinguishing themselves by the versatility in learning long-term serial correlations. Such improvements arise from gates added to the architecture that increase the remembering capacity of the network. LSTM networks were introduced in the seminal work of Hochreiter and Schmidhuber (1997) and later refined by several authors; see Gers, Schmidhuber and Cummins (2000), Gers and Schmidhuber (2001), Zhou et al. (2016), among others.

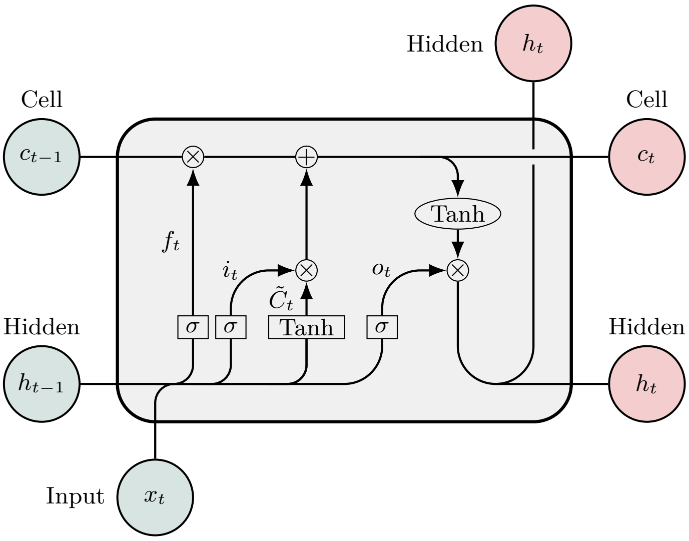


Figure 12 – Basic structure of a LSTM network with a forget gate. Inspired by Fischer and Krauss (2018).

A full description of the operations carried out by a LSTM network with forget gate can be found in the papers by Essien and Giannetti (2019), Yu et al. (2019), and Cao, Li and Li (2019). Briefly, according to the data flow sketched in Figure 12, the LSTM works as follows. Initially, there is a decision about discarding past data, represented by c_{t-1}, which is mathematically determined by the forget function f_t, as it will become clearer later. The function f_t is computed using a sigmoid transformation of hidden and input values such that:

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.100}$$

where w_f are synaptic weights associated with the forget gate and:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.101}$$

In the next stage, new information is generated to be incorporated in the cell state c_t. There are two intermediate steps with the aim of establishing how much information is going to be added, requiring the evaluation of the following expressions:

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.102}$$

$$\tilde{c}_t = \tanh(w_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.103}$$

where w_i and w_c are the synaptic weights associated with the input and the current data. Using these expressions, it is possible to update the previous cell state, c_{t-1}, by combining past and current data:

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{2.104}$$

where $\circ$ denotes the Hadamard product. As one may infer from this equation, f_t and i_t weigh past and current data, respectively, to update the cell state.

The value of the output gate of the LSTM cell is then a scaled and filtered transformation of the inputs and the hidden state given by:

$$o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.105}$$

and the hidden state provided to the next cell of the network is:

$$h_t = o_t \cdot \tanh(c_t) \tag{2.106}$$

An extension of this model is provided by Gers, Schmidhuber and Cummins (2000), who proposed *peephole connections* that allow to take into consideration past and current cell states when estimating the weights i_t and f_t and the output o_t. The mathematical formulation is given by the following system of equations:

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + P_f \cdot c_{t-1} + b_f) \tag{2.107}$$

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + P_i \cdot c_{t-1} + b_i) \tag{2.108}$$

$$\tilde{c}_t = \tanh(w_c \cdot [h_{t-1}, x_t] + P_c \cdot c_{t-1} + b_c) \tag{2.109}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{2.110}$$

$$o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + P_o \cdot c_t + b_o) \tag{2.111}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{2.112}$$

where P_f, P_i and P_o are the peephole weights for the forget gate, input gate, and output gate, respectively.

The solid mathematical foundation and the promising results displayed by LSTM networks in multiple situations are not the only reasons for its widespread use in the literature. O'Reilly and Frank (2006) provide biological justifications to explain the success of these networks, relating the learning mechanisms found in LSTM and similar networks to those present in the prefrontal human cortex.

Lately, diverse applications of LSTM networks have emerged in the pertaining literature. For instance, Messina and Louradour (2015) demonstrate the use of LSTM networks for analysis and comprehension of handwritten sentence. In the field of speech recognition, Li and Wu (2015) and Cai and Liu (2016) offer potential uses, while Fischer and Krauss (2018) and Kim and Won (2018) employ LSTM networks for time series modeling. Another application involves text classification, as shown by Liu and Guo (2019). Greff et al. (2017) provide further examples.

Finally, the thorough investigation executed by Jozefowicz, Zaremba and Sutskever (2015) is noteworthy. These researchers scrutinized the literature, examining over 10,000 different RNN architectures, and identified solely three that could outperform LSTM and GRU in the experiments designed. Even so, the increased performance is not consistent throughout all experimental conditions, implying that these two architectures are competitive against existing alternatives.

2.7 Machine Learning and Inflation Forecasting

Despite the success of machine learning in many fields of research, the literature review carried out shows that applications in Economics have been somewhat neglected. Indisputably, there is an exception, Finance, for which many applications have been conceived in the past years. However, in Macroeconomics, to which the problem of inflation forecasting belongs, fewer empirical studies have been found, regardless of the fact that machine learning applications in Econometrics are fruitful.

An important exception is Medeiros et al. (2019), which were among the first authors to thoroughly survey machine learning models for multi-period inflation forecasting, investigating each one and comparing them in terms of out-of-sample performance. Until then, most authors had focused exclusively on few, simple models, without a thorough comparison between then. For instance, McAdam and McNelis (2005) apply linear and neural network-based "thick" models for forecasting inflation based on Phillips-curve formulations in the US, Japan, and Euro area, concluding that specifications based on thick networks can be competitive against the linear specification.

In a similar fashion, Choudhary and Haider (2012) test ANN models against AR(1) for monthly inflation rates for 28 OECD countries, verifying that ANN models were superior predictors for 45% of the countries of sample, while AR(1) outperformed only in 23%. Combinations of the neural network models also delivered promising results.

Meanwhile, Garcia, Medeiros and Vasconcelos (2017) analyze high-dimensional econometric models, such as shrinkage and complete subset regression, in the context of inflation forecasting in Brazil. The authors obtain compelling results, and conclude that combining forecasts based on model confidence sets can achieve superior predictive performances.

In the case of Medeiros et al. (2019), the authors empirically tested the following approaches using US macroeconomic data to forecast the US CPI:

1. Shrinkage estimators: Ridge regression, LASSO regression, adaLASSO, and elastic net;

2. Factor models: target factors, and factor boosting;

3. Ensemble methods: bagging, complete subset regressions (CSR), and jackknife model averaging (JMA);

4. Random Forests;

5. Hybrid linear-Random Forest models: mixture of Random Forests and ordinary least squares (OLS), and also a combination of RF and adaLASSO; and

6. Other nonlinear models: boosted regression tree, deep neural networks (multi-layered perceptron), polynomial models, and linear models with non-concave penalties.

The performance of each model was checked against three benchmarks: (1) random walk; (2) autoregressive model of order p, or AR(p), where p is determined by the Bayesian Information Criterion (BIC); and (3) unobserved components stochastic volatility (UCSV) models. These benchmarks are comprised by somewhat naive econometric tools that are associated with a linear Phillips Curve and remain popular for forecasting inflation. Their adamant popularity is explained by the difficult to surpass the (weak) performance of these models; see Stock and Watson (1999) and Stock and Watson (2007).

Medeiros et al. (2019) conclude that LASSO, Random Forests and others are able to generate more accurate forecasts than the standard benchmarks, corroborating the benefits of machine learning methods and big data for macroeconomic forecasting. Indeed, the gains are consistent and as large as 30% in terms of mean squared errors. In particular, Random Forests produce the smallest errors and exhibit greater robustness. They also display remarkable stability across different horizons, delivering compelling results in periods of economic expansion and recession as well as in moments of low and high uncertainty. The superiority is also verified in multiple subsamples of the data collected and is robust in real-time experiments.

3 A DEEP LEARNING MODEL FOR INFLATION FORECASTING

This chapter is devoted to the presentation of a novel deep learning model designed for inflation forecasting. With that aim, the theoretical background required for full comprehension is discussed, focusing on convolutional LSTM networks and variational autoencoders. Subsequently, the combination of these techniques and the computational implementation are explored.

3.1 Theoretical Background

Recently, several empirical papers have reported encouraging results regarding the use of deep learning techniques in problems involving time series forecasting. This success is explained mainly by the fact that neural networks, as universal approximators in many situations, are capable of capturing and dealing with nonlinearities, enhancing their flexibility. Some studies illustrating the power and versatility of those models are compiled in Table 1. The interested reader may find exhaustive surveys on the subject in the papers by Ahmed et al. (2010), Atsalakis and Valavanis (2009), Längkvist, Karlsson and Loutfi (2017), and Liu et al. (2017).

Table 1 – Empirical papers on applications of deep learning techniques for time series forecasting.

Authors	**Model**	**Application**
Adamowski (2008)	MLP	Water demand
Galeshchuk (2016)	MLP	Exchange rates
Kim and Won (2018)	LSTM + GARCH	Stock index volatility
Kim et al. (2004)	ANN	Non-stationary time series
Shi et al. (2015)	ConvLSTM	Rainfall
Wang, Qi and Liu (2019)	ConvLSTM	Energy generation

Motivated by these results, a deep learning model for inflation forecasting will be developed in this section. The central idea is to build on the adaptability and robustness of LSTM networks in capturing temporal dependencies to formulate a model that is well-suited to address the nonlinearities observed in inflation time series. Simultaneously, convolutional networks are added to take advantage of hierarchical patterns that may exist in data, together with autoencoders for dimension reduction. Figure 13 depicts the flowchart of the forecasting process. The core references that inspired the development of this model are Bao, Yue and Rao (2017), Essien and Giannetti (2019), Shi et al. (2015) and Wang, Qi and Liu (2019). In these studies, the authors have implemented models combining

LSTM and convolutional networks, obtaining positive results in terms of out-of-sample performance when compared to simpler, more conventional models.

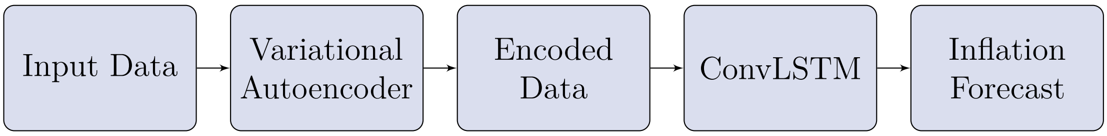


Figure 13 – Flowchart exhibiting the proposed forecasting process.

Briefly, the model developed herein is represented by a convolutional LSTM network whose inputs are provided by the encoding layer of a variational autoencoder. As of the date of publication of this dissertation, to our knowledge, no paper has investigated and applied convolutional LSTM networks and variational autoencoders together to forecast macroeconomic variables. Evidently, many papers study these techniques individually or coupled with other networks. However, due to the fact that these networks have only been recently introduced in the literature, there is an opportunity to examine their interaction in the context of time series forecasting. Consequently, it is reasonable to claim that a contribution to the literature is the assessment of the performance of this combination.

3.1.1 Variational Autoencoders

In the architecture proposed, the *variational autoencoder* (VAE) serves to reduce the dimension of the input data, since the dataset provided by McCracken and Ng (2016), which is employed to fit the ConvLSTM model, contains several, highly correlated time series, implying that some redundancy is expected. As an illustration of the relevance of dimension reduction, in a real time forecasting exercise, Boivin and Ng (2006) find that factors extracted from as few as 40 pre-screened series often yield satisfactory or even better results than using all the 147 series available in that occasion. Besides, weighting the data based on their properties when conceiving the factors also lead to more accurate forecasts, and VAEs have been successfully applied to transform, encode, and extract features from time series, as shown by Pereira and Silveira (2018), among others.

Therefore, the architecture begins with a variational autoencoder acting analogously to a nonlinear PCA, mixing the original series and producing a smaller dataset that can be used to model and forecast inflation. The comparison is justified by the fact that, similarly to the conventional PCA, autoencoders allow to extract principal components by encoding data for later decoding with the intent of replicating the original data. The nonlinear aspect stems from the fact that, unlike PCA, the components found are not linearly uncorrelated. Such feature is not a drawback, for neural networks are designed to handle and learn from these nonlinearities.

With respect to the mathematical formulation of variational autoencoders, following Doersch (2016), the foundations of VAEs has actually little to do with classical autoencoders such as sparse or denoising ones. The name is justified by the fact that the encoding-decoding structure is also available here, resembling a traditional autoencoder. However, unlike former versions, commonly no tuning parameter analogous to sparsity penalties is incorporated. In addition, contrasting with sparse and denoising autoencoders, samples can be generated without performing MCMC. Even though they share these encoding-decoding abilities with other versions of autoenconders, they are more closely related to generative networks. In fact, they can be regarded as one of the first generative networks introduced in the pertaining literature.

Essentially, a VAE learns stochastic mappings between an observed dataset X, whose distribution is complicated and unknown, and a latent set of variables z whose distribution is simple and, therefore, from which samples may be generated (KINGMA; WEILLING, 2019). Formally, it is assumed that the generative model of a given dataset X depends on a vector of latent variables z in a high-dimensional space $\mathcal{Z}$ obeying some probability density function $p(z)$ defined over $\mathcal{Z}$. Next, assume there is a family of deterministic functions $f(z;\theta)$ parameterized by a vector $\theta \in \Theta$ such that $f : \mathcal{Z} \times \Theta \to \mathcal{X}$. The problem is to optimize θ such that, by sampling z from $p(z)$, with high probability, $f(z;\theta)$ will generate samples from the population of X. As put by Doersch (2016), we are aiming to maximize the probability of each X in the training set under the entire generative process, according to:

$$p(X) = \int p(X|z;\theta)p(z)dz \tag{3.1}$$

where $f(z;\theta)$ has been replaced by $p(X|z;\theta)$ to allow the use of the law of total probability. In Bayesian Statistics, $p(z)$ is known as the prior distribution over the latent space.

In this framework, variational autoencoders can be trained to generate the desired samples, as shown in Goodfellow, Bengio and Courville (2016). With this purpose, the VAE initially draws a sample z from the distribution $p(z)$, which is run through a differentiable generator network $f(z;\theta)$. Comparing with the previous equation, f may be seem as a mapping that determines how z and $p(X|z;\theta)$ are connected, where θ represents here the parameters of the generator network. Finally, X is sampled from a distribution $p(X;f(z)) = p(X|z)$, where θ is omitted for simplicity. Nevertheless, during training, the approximate inference network (or encoder) $q(z|X)$ is used to obtain z and $p(X|z)$ is then viewed as a decoder network. Note that $q(z|X)$ approximates the true, but intractable posterior of the generative model; see Kingma and Weilling (2019).

Regarding the choice of $p(X|z)$, Gaussian distributions are often used, i.e.:

$$p(X|z) = N(X|f(z;\theta)) \tag{3.2}$$

where $N(\mu,\Sigma)$ denotes a multivariate normal distribution with mean μ and covariance matrix Σ. As contended by Doersch (2016), the Gaussian distribution allows the use of gradient descent (or any other optimization technique) to increase $p(X)$ by forcing $f(z;\theta)$ to approach X for some z, i.e., progressively rendering the training data more likely under the generative model. Naturally, other distributions are feasible, as long as they are continuous in θ.

For illustration, Figure 14 outlines the general architecture of a variational autoencoder. Notoriously, VAEs distinguish themselves by the generative (sampling) procedure that occurs immediately after the inputs are encoded. Comparing with Equation 3.2, we see that the effect is that the encoded data is the input of $f(z;\theta)$, which is itself a neural network dedicated for the estimation of the parameter θ. For a Gaussian, θ is simply the mean μ and the covariance matrix Σ of the distribution.

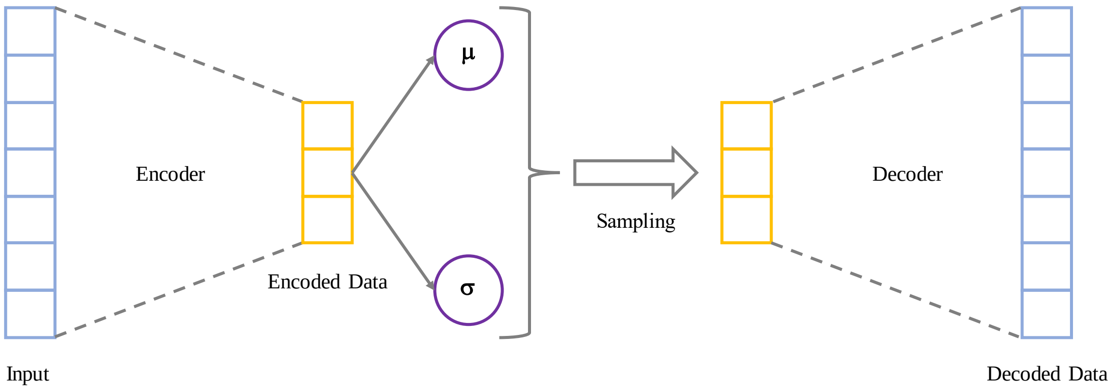


Figure 14 – Representation of a variational autoencoder. In the sampling stage, μ and σ denoted the mean and standard deviation estimated from the encoded data, respectively.

As shown by Goodfellow, Bengio and Courville (2016), VAEs are trained by maximizing the variational lower bound $\mathcal{L}$ associated with data point x:

$$\mathcal{L}(q) = E_{z\sim q(z|X)}\left[\ln p(z,X)\right] + \mathcal{H}(q(z|x)) \tag{3.3}$$

In this expression, the first term is the joint log-likelihood of the visible and hidden variables under the approximate posterior over the latent variables. Meanwhile, $\mathcal{H}$ is the entropy of the approximate posterior. In practice, the effect of the entropy is that it stimulates the variational posterior to place high probability mass on many z values that could have generated x, rather than collapsing to a single point estimate of the most likely value.

Manipulating Equation 3.3, we may write:

$$\mathcal{L}(q) = E_{z\sim q(z|X)} \ln p(X|z) - D_{KL}(q(z|X)||p(z)) \leq \ln p(x) \tag{3.4}$$

where D_{KL} is the *Kullback-Leibler (KL) distance*, also known as *relative entropy*. For any distributions P and Q of a continuous random variable, D_{KL} is defined to be the integral:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \ln\left(\frac{p(x)}{q(x)}\right) dx \tag{3.5}$$

where p and q denote the probability densities of P and Q, respectively. More generally, if P and Q are taken to be probability measures over a set $\mathcal{X}$, and P is absolutely continuous with respect to Q, then:

$$D_{KL}(P||Q) = \int_{\mathcal{X}} \ln\left(\frac{dP}{dQ}\right) dP \tag{3.6}$$

where dP/dQ is the Radon-Nikodym derivative of P with respect to Q. Equation 3.4 shows that $\mathcal{L}(q)$ is the sum of the log-likelihood with the KL distance between the posterior and the model prior. Inference and learning can be carried out using traditional gradient-based optimization algorithms; see Kingma and Weilling (2019).

3.1.2 Convolutional LSTM Networks

The next step involves the convolutional LSTM (ConvLSTM henceforth) network, which is the core of the forecasting process. The choice for this extension over the traditional LSTM network is justified by the fact that, although the latter has proven powerful for handling temporal serial correlation, it contains too much redundancy for spatial data, as asserted by Shi et al. (2015), who are credited for the development of ConvLSTM. This drawback is addressed by adding convolutional structures in the input-to-state and the state-to-state transitions. Therefore, the input and recurrent transformations are both convolutional in a ConvLSTM layer.

Mathematically, a *ConvLSTM network* can be formulated as:

$$i_t = \sigma(w_{xi} * x_t + w_{hi} * h_{t-1} + w_{ci} \circ c_{t-1} + b_i) \tag{3.7}$$

$$f_t = \sigma(w_{xf} * x_t + w_{hf} * h_{t-1} + w_{cf} \circ c_{t-1} + b_f) \tag{3.8}$$

$$o_t = \sigma(w_{xo} * x_t + w_{ho} * h_{t-1} + w_{co} \circ c_{t-1} + b_o) \tag{3.9}$$

$$\tilde{c}_t = \tanh(w_{xc} * x_t + w_{hc} * h_{t-1} + b_c) \tag{3.10}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{3.11}$$

$$h_t = o_t \circ \tanh(c_t) \tag{3.12}$$

where σ is the sigmoid function, and w_{xi}, w_{xf}, w_{xo}, w_{xc}, w_{hi}, w_{hf}, w_{ho} and w_{hc} are 2D convolution kernels. The input x_t, the cell state c_t, the hidden state h_t, the forget gate f_t, the input gate i_t, and the output gate o_t are all 3D tensors. The symbol $*$ denotes the convolution operator, and $\circ$ is the Hadamard product. Comparing this formulation with the equations derived in subsection 2.6.6, one may observe that the main difference between a conventional LSTM network and ConvLSTM lies on the convolutions carried out when updating states and gates. Experiments show that ConvLSTM networks are more effective at capturing spatiotemporal correlations, consistently outperforming standard LSTM networks; see Shi et al. (2015).

It is worth emphasizing that the choice of a hybrid model that merges convolutional and LSTM networks is not only supported by the empirical results reported in the literature regarding the successful applications of ConvLSTM for time series modeling and forecasting. Supplementary studies show that, in general, hybrid models tend to deliver more robust performance than a single model in several situations. For instance, Bai, Tang and An (2019) demonstrate the effectiveness of utilizing CNNs and LSTMs to classify scene images with multi-views and multi-levels of abstraction. This combination outperforms several state-of-the-art methods in their experiments. In distinct contexts, similar results are provided by Kristjanpoller and Minutolo (2015), who merge ANNs and GARCH models to predict the price return volatility of gold spot and future prices; by Dash et al. (2010), who combine genetic algorithms and ANNs for groundwater level prediction; and by Khashei and Bijari (2011), whose model for time series forecasting stems from a hybridization of artificial neural networks and ARIMA models.

Moreover, with the accelerated development and gradual maturity of deep learning, some practitioners began to realize that the local connection and global sharing features of convolutional neural networks can greatly diminish the required parameters and training time of the model. This approach is applied by Sezer and Ozbayoglu (2018) for financial trading, where the authors select 15 distinct technical indicators and compute their values for a 15 day period to convert price series into 2D images, which are later processed by a deep CNN.

As another example, Lee and Kim (2020) employ ConvLSTM, trend sampling, and specialized data augmentation for stock market forecasting. The framework developed can successfully learn high-level features from super-high dimensional time series. Specific regularization and mini-batch sampling techniques are used to improve out-of-sample performance. In an application involving the stock indices S&P500, KOSPI200, and FTSE100, the model managed to defeat its competitors. The work highlights not only the advantages of merging several approaches for time series forecasting, but also the importance of techniques focused on preventing overfitting, which are discussed subsequently.

3.2 Computational Implementation

After establishing the theoretical foundations of variational autoencoders and convolutional LSTM networks, it is feasible to proceed and commence the explanation of the model proposed herein for inflation forecasting. In addition, the computational implementation can also be approached. With this intent, Figure 15, Figure 16, and Figure 17, which portray the VAE and the ConvLSTM networks underpinning the model, will orient the discussion.

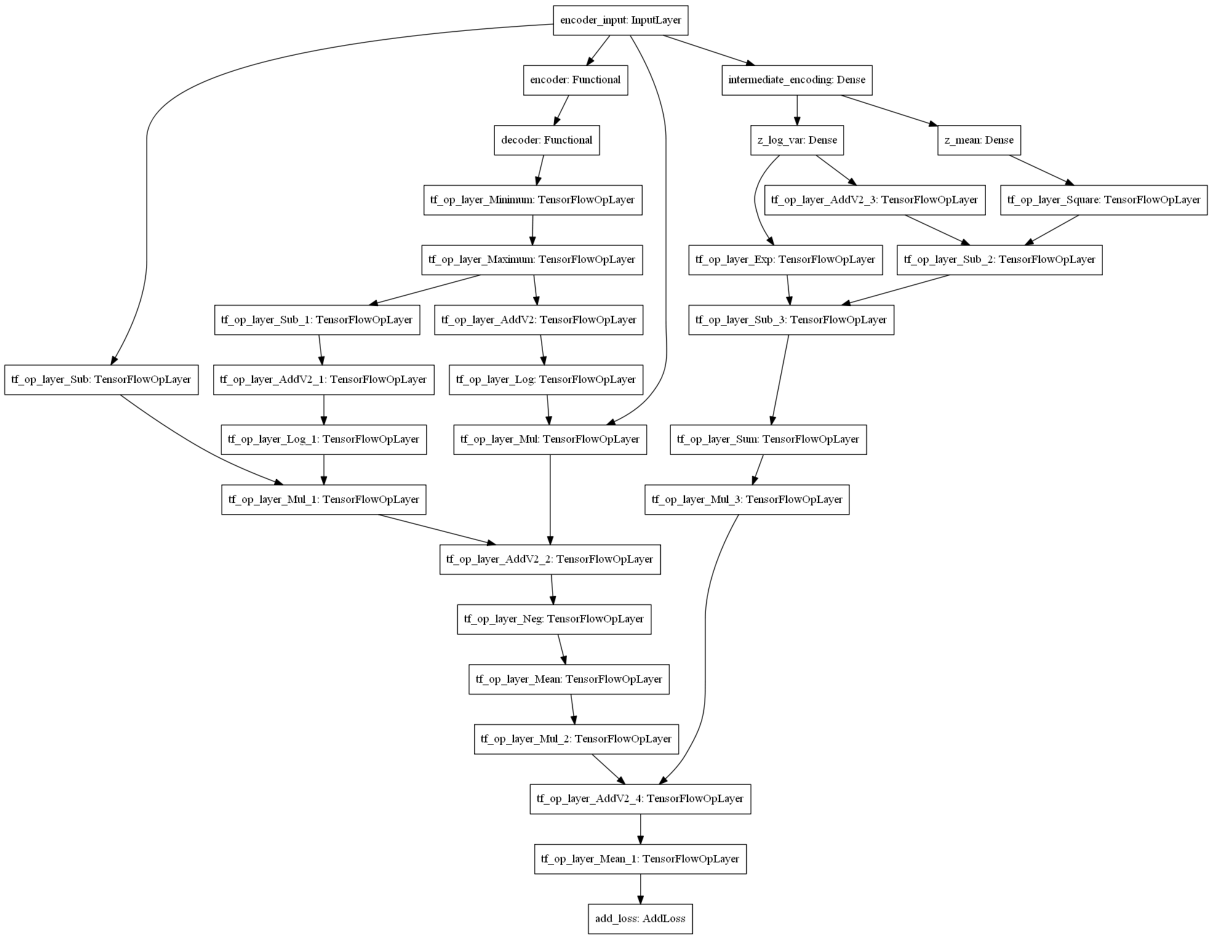

encoder_input: InputLayer
encoder: Functional
intermediate_encoding: Dense
decoder: Functional
z_log_var: Dense
z_mean: Dense
tf_op_layer_Minimum: TensorFlowOpLayer
tf_op_layer_AddV2_3: TensorFlowOpLayer
tf_op_layer_Square: TensorFlowOpLayer
tf_op_layer_Maximum: TensorFlowOpLayer
tf_op_layer_Exp: TensorFlowOpLayer
tf_op_layer_Sub_2: TensorFlowOpLayer
tf_op_layer_Sub_1: TensorFlowOpLayer
tf_op_layer_AddV2: TensorFlowOpLayer
tf_op_layer_Sub_3: TensorFlowOpLayer
tf_op_layer_Sub: TensorFlowOpLayer
tf_op_layer_AddV2_1: TensorFlowOpLayer
tf_op_layer_Log: TensorFlowOpLayer
tf_op_layer_Log_1: TensorFlowOpLayer
tf_op_layer_Mul: TensorFlowOpLayer
tf_op_layer_Sum: TensorFlowOpLayer
tf_op_layer_Mul_1: TensorFlowOpLayer
tf_op_layer_Mul_3: TensorFlowOpLayer
tf_op_layer_AddV2_2: TensorFlowOpLayer
tf_op_layer_Neg: TensorFlowOpLayer
tf_op_layer_Mean: TensorFlowOpLayer
tf_op_layer_Mul_2: TensorFlowOpLayer
tf_op_layer_AddV2_4: TensorFlowOpLayer
tf_op_layer_Mean_1: TensorFlowOpLayer
add_loss: AddLoss

Figure 15 – Flowchart representing the implementation of the variational autoencoder using the library Keras in Python. Even though this characterization is far more complex than the one outlined in Figure 14, the encoding, decoding, and sampling elements are still present. The additional layers are simply intermediate functions required by Keras for implementation.

According to Figure 13, in the first stage, the input data is loaded in the variational autoencoder, whose implementation is depicted in Figure 15. Fundamentally, the architecture proposed is closely related to the one examined in previous sections. Indeed, for $p(X|z)$, a normal distribution is selected. The mean vector and covariance matrix are estimated via a fully connected MLP that receives as input the encoded data. The strategy of maximizing the variational lower bound is also employed here.

Moving forward, the encoded data conceived by the VAE is transferred to the ConvLSTM network represented in Figure 16, whose implementation in Python is shown in Figure 17. The flowcharts unveil the inner layers of the architecture, displaying the existence of:

1. Two ConvLSTM layers, which are responsible for extracting spatiotemporal features

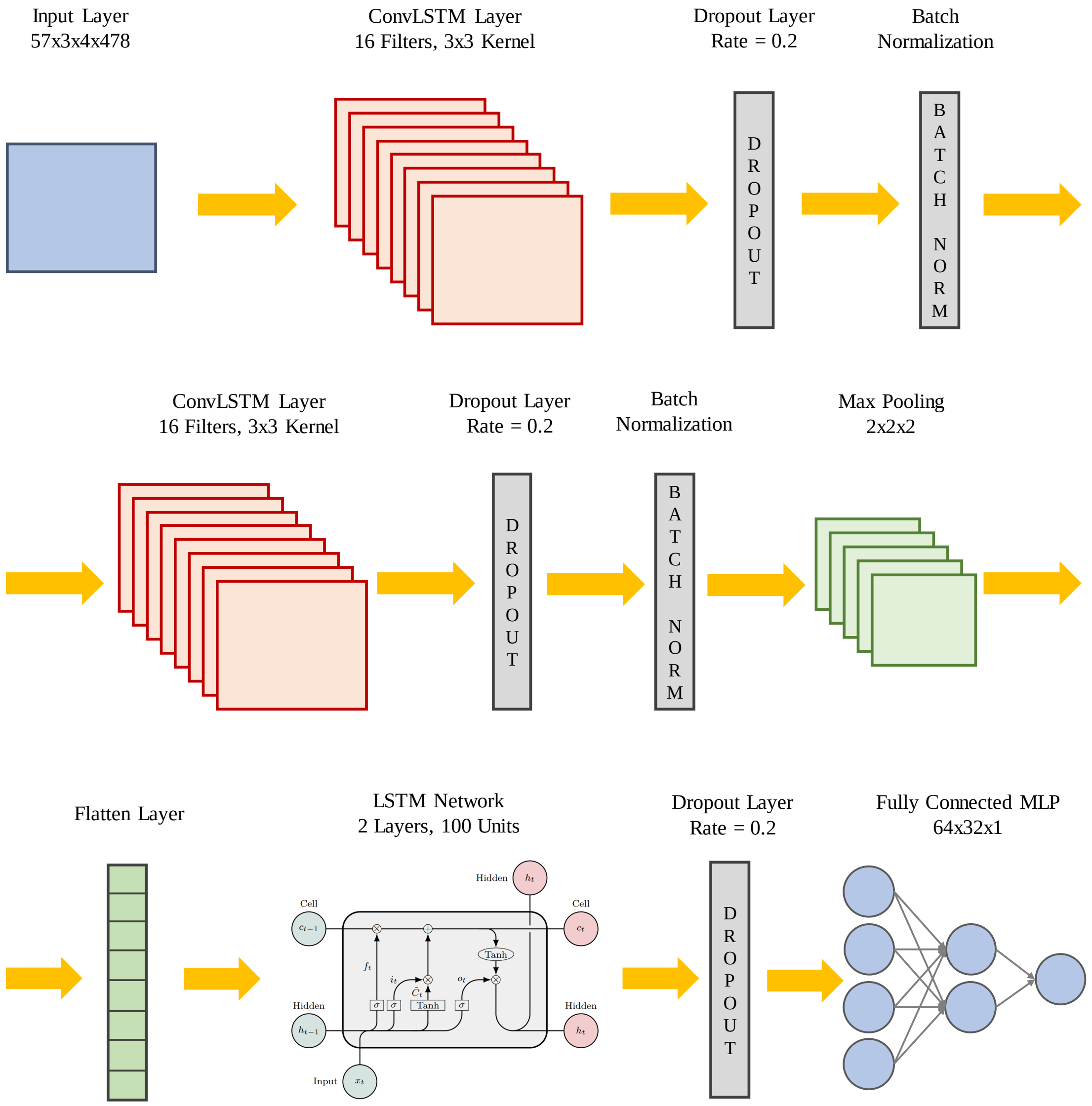


Figure 16 – Flowchart representing the layers of the ConvLSTM model designed in this study. The input layer receives the encoded data supplied by the variational autoencoder (a 57×3×4×478 tensor) and transfers the data to a sequence of convolutional LSTM, dropout (with a dropout rate of 0.2), and batch normalization layers. The convolutional layers have 16 filters and a 3×3 kernel (filter size). Next, a 3D max pooling layer (2×2×2) summarizes the data, which is then flattened and inserted in a LSTM network formed by two layers. with 100 units each, whose output is processed by a fully connected deep multilayer perceptron (constituted by 3 layers, with 64, 32 and 1 units each, respectively). Finally, the deep MLP generates inflation forecasts.

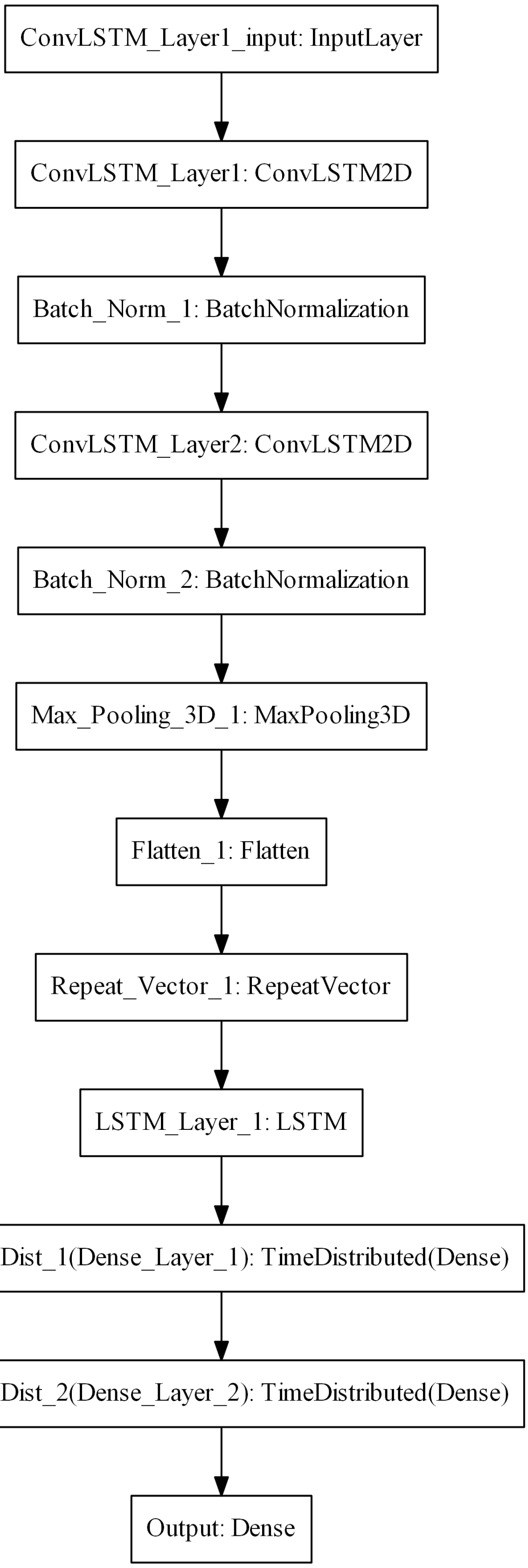

ConvLSTM_Layer1_input: InputLayer
ConvLSTM_Layer1: ConvLSTM2D
Batch_Norm_1: BatchNormalization
ConvLSTM_Layer2: ConvLSTM2D
Batch_Norm_2: BatchNormalization
Max_Pooling_3D_1: MaxPooling3D
Flatten_1: Flatten
Repeat_Vector_1: RepeatVector
LSTM_Layer_1: LSTM
Time_Dist_1(Dense_Layer_1): TimeDistributed(Dense)
Time_Dist_2(Dense_Layer_2): TimeDistributed(Dense)
Output: Dense

Figure 17 – Flowchart representing the implementation of the advocated ConvLSTM model using the library Keras in Python. The steps are consistent with those depicted in Figure 16. Dropout is embedded in the ConvLSTM layers.

and transforming the time series, generating almost a timeline that shows when different features appear in the time series. Each of these are immediately followed by dropout and batch normalization layers, whose roles will be scrutinized subsequently;

2. A summarizing max pooling layer that receives as input the outputs of the convolutional layers;

3. A flatten layer that converts tensors into vectors;

4. A conventional, simple, stacked LSTM network that receives the flattened tensors and captures temporal dependencies. These LSTM layers also encompass dropout layers; and

5. A MLP that reduces the output of the LSTM network to a single inflation forecast.

Thus, the network begins effectively with two ConvLSTM layers intertwined with dropout and batch normalization layers, which receive the encoded data from the VAE in the form of a 4D tensor. Each dimension of the tensor is related to a different aspect of the data: (a) the number of “principal components” generated by the VAE; (b) the number of observations in the period analyzed; (c) the number of subsets into which each window of observations is split; and (d) the number of elements per subset. This approach resembles the one proposed by Sezer and Ozbayoglu (2018) to convert time series into “images” that can be processed by convolutional networks, including the ConvLSTM.

As one might expect, various configurations were subject to evaluation until the tensor described in Figure 16 was reached. This empirical approach was deemed satisfactory due to the fact that, being ConvLSTM networks a emerging field, no consensus regarding the optimal configuration exists, and, given the peculiarities of every problem feasible to tackle with these networks, a general rule may never be conceived. The thorough investigation of more effective architectures is reserved for future studies, since the objective here is to prove that these networks can be successfully wielded to model and forecast time series, in particular those originated by macroeconomic variables.

Furthermore, the ConvLSTM presents the same structure proposed by Shi et al. (2015), which is described by Equation 3.7 to Equation 3.12. Concerning the configuration of this layer, the activation function chosen is the SELU function developed by Klambauer et al. (2017) due to its superior properties in comparison to other options such as sigmoid or ReLU. Initialization of the neurons is based on drawing the weights from a truncated Gaussian distribution; see Klambauer et al. (2017).

Inspired by the configurations tested by Sezer and Ozbayoglu (2018), 16 filters and a filter (kernel) size of 3×3 provide the convolution capability with closest neighbors’ (upper, lower, right, left, upper left, upper right, lower left, and lower right) information while processing the current layer. Thus, sharp variations within the matrix can be captured and a decent number of lags will be assessed each time the filter is run. Also, adhering to the best practices in the literature, zero padding is adopted, meaning that the size of the input is automatically adjusted to avoid shrinking the spatial extent of the network rapidly and/or using small kernels, harming the generalization power of the network; see Goodfellow, Bengio and Courville (2016).

It is worth mentioning that, although there is no clear rule in the literature guiding how to optimize the filter size, few filters cannot infer enough features to improve the learning ability of the network. On the other hand, an excessive number of filters may decrease the efficiency of the feature extraction process, since too much data will be analyzed simultaneously and, thus, the filter may become blurry and fail to identify the underlying features.

At this point, it should be highlighted that neural networks are known not to be very robust to noise, as claimed by Tang and Eliasmith (2010). Furthermore, overfitting is a serious risk in such models, as argued by Srivastava et al. (2014). Bishop (2006) proposes *regularization* as an approach to controlling the complexity of a model, focusing on improving the generalization power of a neural network. Generically, regularization is any modification made to a learning algorithm with the intent of reducing the generalization error, but not necessarily its training error (GOODFELLOW; BENGIO; COURVILLE, 2016).

Hence, an alternative to circumvent the aforementioned limitations of neural networks is the addition of *dropout* layers, which is a regularization strategy that can be seen as a procedure of constructing new inputs by multiplying by noise (GOODFELLOW; BENGIO; COURVILLE, 2016). Dropout provides a computationally inexpensive, but potent method of regularizing a broad family of models (GOODFELLOW; BENGIO; COURVILLE, 2016). Moreover, training is virtually unchanged, for dropout networks can be trained using stochastic gradient descent in a similar fashion to standard networks.

The main idea behind dropout is to randomly drop units, along with their connections, from the network during the training stage, preventing exaggerated co-adaption, as advocated by Srivastava et al. (2014); see Figure 18. In some sense, it resembles bagging, but with an exponentially large number of neural networks acting as the different models used in bagging (GOODFELLOW; BENGIO; COURVILLE, 2016). Dropout significantly reduces overfitting and is competitive against other similar strategies. Indeed, Srivastava et al. (2014) report performance improvement on supervised learning tasks in vision, speech recognition, document classification and computational biology.

Mathematically, inspired by Srivastava et al. (2014), dropout can be formulated as follows. Consider a standard feedforward network with L hidden layers and let $1, \ldots, L$ be the index of these layers and $l \in \{0, \ldots, L-1\}$. Also, let $\mathbf{z}^{(l)}$ denote the vector of inputs $z_i^{(l)}$ into layer l, and $\mathbf{y}^{(l)}$ be the vector of outputs $y_i^{(l)}$ of this layer. In addition, $W^{(l)}$ and $\mathbf{b}^{(l)}$ are the matrix of weights $\mathbf{w}_i^{(l+1)}$ and the vector of biases $b_i^{(l+1)}$ at layer l. The

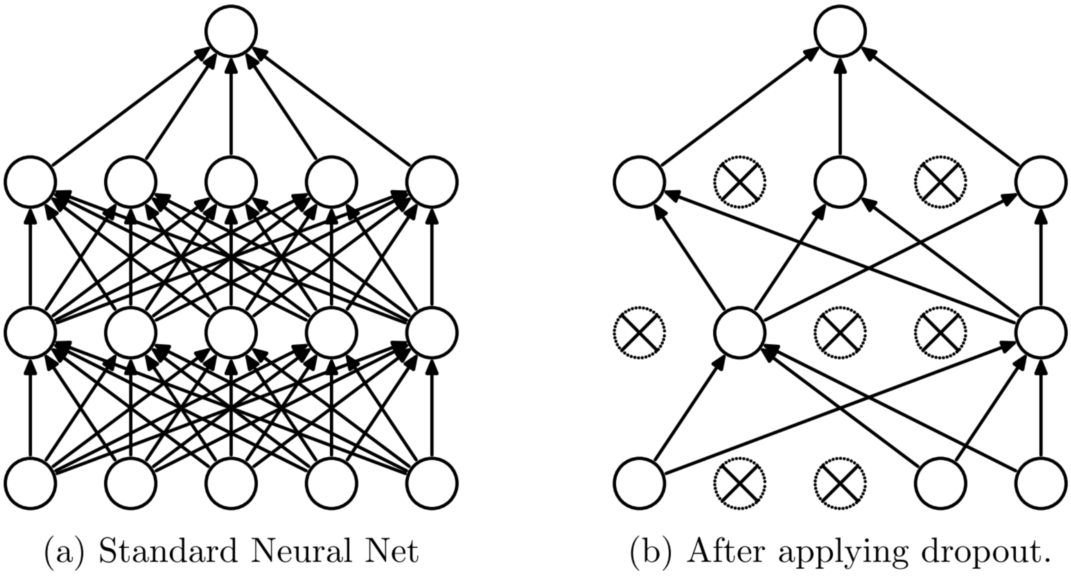


Figure 18 – Illustration of the possible impacts of dropout in a standard neural network model. In the left, there is a fully connected MLP with two hidden layers. In the right, after the dropout procedure, units have been dropped and the network loses several connections. Source: Srivastava et al. (2014).

operations that occur at the neurons i of every layer l of the network are:

$$z_i^{(l+1)} = \mathbf{w}_i^{(l+1)} \cdot \mathbf{y}^{(l)} + b_i^{(l+1)} \tag{3.13}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)}) \tag{3.14}$$

where f is a predefined activation function.

When dropout is added, the operations become:

$$r_j^{(l)} \sim Bernoulli(p) \tag{3.15}$$

$$\tilde{\mathbf{y}}^{(l)} = \mathbf{r}^{(l)} \circ \mathbf{y}^{(l)} \tag{3.16}$$

$$z_i^{(l+1)} = \mathbf{w}_i^{(l+1)} \cdot \tilde{\mathbf{y}}^{(l)} + b_i^{(l+1)} \tag{3.17}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)}) \tag{3.18}$$

where $\circ$ is the element-wise product, and $r_j^{(l)}$ is a vector of independent Bernoulli random variables with probability p of being 1 and $1-p$ of being 0. In every iteration, this vector is randomly generated and multiplies $\mathbf{y}^{(l)}$ to create thinned outputs. The result is equivalent to sampling a sub-network from a larger network. During the test window, weights are scaled as $W_{test}^{(l)} = pW^{(l)}$.

In the model proposed for inflation forecasting, dropout layers have been added exclusively after the ConvLSTM and the LSTM layers. This decision is justified by the fact that the core of the forecasting process happens at those stages. The succeeding MLP layers essentially convert their output into a inflation estimate, but the knowledge is concentrated in the previous engines. However, to test this reasoning, other configurations have been implemented, with additional dropout layers, and the results were virtually the same in terms of out-of-sample performance. Moreover, the calibration of the parameter p, which was set in the final version of the model as $p = 0.8$ or, equivalently, a dropout rate of 0.2, was carried out empirically, since there is no consensus in the literature and the optimal choice varies according to the context considered.

Another way to improve optimization and out-of-sample performance in deep neural networks is *batch normalization*, which reparameterizes the network by inserting both additive and multiplicative noise on hidden units at training time. The reparametrization significantly attenuates the problem of coordinating updates across many layers (GOODFELLOW; BENGIO; COURVILLE, 2016). Essentially, batch normalization involves normalizing the data flowing through the layers of a network, i.e., ensuring that, from a layer to the next, data has zero mean and unit standard deviation.

The main implication is that gradient-based optimization algorithms become immune to operations that simply change the standard deviation or the mean of the data without effectively improving the out-of-sample performance of the model (GOODFELLOW; BENGIO; COURVILLE, 2016). Precisely, according to the empirical analysis provided by Cho et al. (2014), networks without batch normalization are prone to suffering from learning limitations, since large gradient updates may result in diverging loss and activations growing exponentially with network depth. Santurkar et al. (2018) demonstrates that the improvements obtained from batch normalization emerge from the fact that this strategy makes the optimization landscape smoother, and this smoothness induces a more predictive and stable behavior of gradients, allowing for faster training. In the simulations performed in this study, batch normalization had a positive effect, consistently with the results reported by Ioffe and Szegedy (2015).

Technically, for a layer with d-dimensional input $x = (x^{(1)}, \ldots, x^{(d)})$, each dimension is normalized via the function (IOFFE; SZEGEDY, 2015):

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sigma[x^{(d)}]} \tag{3.19}$$

where the expectation $E[\cdot]$ and standard deviation $\sigma[\cdot]$ operators are computed conditioned on the training set. It is worth noting that, for convolutional layers, ConvLSTM layers included, normalization must obey the convolutional property, as observed by Ioffe and Szegedy (2015). Equivalently, different elements of the same feature map, in distinct locations, must be subject to the same normalization process. In order to achieve this, Ioffe and Szegedy (2015) advocate the joint normalization of all activations in a mini-batch, over all locations.

The model designed also contains a pooling layer. By definition, its purpose is to transform the output of the previous layer prior to letting the data continue to flow through the network. In practical terms, the pooling function replaces the output at a certain location with a summary statistic computed using nearby outputs. The consequence is that the representation becomes almost invariant to small translations of the input, which is a desirable property whenever the presence of a certain feature is more important than its position in the data. As highlighted by Goodfellow, Bengio and Courville (2016), the use of pooling can be equivalently viewed as adding an infinitely strong prior that the function the layer learns must be invariant to small translations.

A simple way to understand how pooling works is to consider a hypothetical neural network designed to identify whether a sequence of images contains dogs. Obviously, the position of a dog in each image is irrelevant, and the model must only be concerned with its presence. Thus, the network should be trained with this objective, and pooling layers give significant contributions. In the case of time series, these layers can be useful to identify extreme events (i.e. shocks), for instance. It is not necessary to known the exact moment when a shock occurs, solely to confirm that, in a certain time window, it might have taken place and, therefore, the observations in this period might not be representative of the long-term trends.

One of the most common types is the *max pooling* function developed by Zhou and Chellappa (1988). In its most basic form, the function computes the maximum output within a rectangular neighborhood (GOODFELLOW; BENGIO; COURVILLE, 2016). Scherer and Müller (2010), in a study devoted to convolutional architectures for object recognition, focus on gaining insights into distinct pooling functions by directly comparing them via a common architecture for a wide selection of object recognition tasks. The experiments reveal that a maximum pooling operation significantly outperforms the competitors. Corroborating these findings, Murray and Perronnin (2014) demonstrate that a generalized form of max pooling leads to significant performance gains with respect to heuristic alternatives in five public image classification benchmarks.

The ensuing layer found in the model proposed is a flatten layer, which performs a mathematical operation analogous to the vectorization of matrices. In the code implemented, the flatten layer converts tensors into arrays. The operation is required for the next step, involving stacked LSTMs. The LSTM model adopted here is similar to the version with a forget gate advocated by Gers, Schmidhuber and Cummins (2000) and implemented by Bao, Yue and Rao (2017) and Fischer and Krauss (2018). A total number of 100 cells is adopted in each layer, and dropout layers with $p = 0.8$ are also added. The activation function and the recurrent function selected are hyperbolic tangent and sigmoid, respectively, choices which have been guided by the best practices found in the literature.

Naturally, other configurations of LSTM cells could have been implemented. In fact, some of these possibilities were tested, although, for the sake of objectivity, the results have been omitted. An example is the number of cells in the layer. Increasing or decreasing the number with respect to the base case seemed to have little influence on the out-of-sample performance. Additionally, the number of LSTM layers also had weak effects. Other popular versions of LSTM, such as the Bidirectional LSTM (see Yu et al. (2019) for details), were also examined, without delivering a statistically superior performance. Nevertheless, a thorough scrutiny of these architectures, focusing on their impacts on the performance of the model, is desirable and could be the subject of future studies.

In the sequence, a fully-connected MLP converts the output of the LSTM into a inflation forecast. The layer starts with a hidden layer of 64 units, which is linked to another hidden layer comprised by 32 units and, finally, to an output neuron. Consistently with preceding choices, SELU is the activation function of each of these neurons, even though, in this part of the model, there is a marginal difference in performance with respect to the scenario in which ReLU is employed. Again, other configurations are conceivable. For instance, the number of hidden layers and of units per layer could be modified. However, the final accuracy is not significantly sensitive to these choices.

4 DATA AND METHODOLOGY

In this section, we discuss the data and methodology for fitting the model created for inflation forecasting and the benchmarks elected for performance evaluation. At the outset, these benchmarks are introduced, emphasizing their implementation for posterior empirical assessment of the out-of-sample predictive accuracy. In the sequence, the dataset is presented, also explaining the breakdown of the samples into training and test ones. Finally, the programming languages, libraries, and functions required for the computational routines used to fit the models are examined as well.

4.1 Benchmarks for Performance Appraisal

A broad, diverse, and fundamentally justifiable collection of benchmarks is crucial for the appraisal of forecasting models. Without question, selecting inadequate benchmarks may result in a misleading gauge of the out-of-sample precision of a model, since accuracy analysis must always be executed in a relative basis, comparing against competing alternatives for prediction. Indeed, the primary concern is to establish whether the model developed is capable of reducing the forecasting errors encountered in models commonly applied by practitioners and academics.

In an effort to accomplish this objective, a set of 25 benchmarks has been elected for relative performance analysis, which are listed below. The details of those that requiring the tuning of hyperparameters are examined afterwards.

1. Random walk;
2. Ridge regression (coupled with cross-validation).
3. Bayesian Ridge regression;
4. LASSO regression (coupled with cross-validation);
5. Bayesian LASSO;
6. Elastic net;
7. Support vector regression (coupled with PCA for dimension reduction);
8. Random Forests (coupled with PCA for dimension reduction);
9. BART (coupled with PCA for dimension reduction);
10. Bagging (coupled with PCA for dimension reduction);
11. K-Nearest Neighbors regression (coupled with PCA for dimension reduction);

12. Robust regression with Huber loss;
13. Theil-Sen robust regression;
14. Factor models;
15. GARCH model;
16. Vector error correction model (VECM);
17. SETAR model;
18. Moving average model, based on Atkeson and Ohanian (2001);
19. Seasonal ARIMA (SARIMA) model;
20. ARFIMA (fractional ARIMA);
21. Gradient boosting;
22. AdaBoost;
23. Bayesian regression;
24. Multilayer perceptron;
25. Standard LSTM network.

The random walk model is the most naive forecasting model. It simple assumes that one-period-ahead expected inflation, conditional on all the past observations, equals the rate observed in the most recent period. As a stochastic process, it behaves as a discrete-time martingale, implying that no insight can be extracted using previous inflation rates and/or other variables. Thereby, due to its simplicity and assumptions, it is a natural benchmark to determine whether it is feasible to generate more accurate predictions using more complex model to embody past and current data and relate to one-period-ahead inflation.

Ridge regression and its Bayesian version are implemented exactly according to the exposition in subsection 2.3.2. In the first case, cross-validation is adopted to estimate the optimal relative weight between regulation and goodness-of-fit. The inputs are not treated in any way, meaning that no dimension reduction and denoising techniques are used here. Indeed, the Ridge estimator, by penalizing the sum of squared coefficients, attempts to reinforce significant variables while shrinking those deemed unnecessary. In the process, it also mitigates the effects of multicollinearity.

The implementation of LASSO regression follows subsection 2.3.1 and, analogously to the Ridge regression implementation, cross-validation is employed to estimate the hyperparameter. The same is valid for Elastic Net, covered in subsection 2.3.3. Moreover, the Bayesian LASSO regression was implemented following Park and Casella (2008) and the manual of the library *monomvn* in R[1]. The main difference is that, for improved mixing, a Rao-Blackwellized sample of σ^2 (with β integrated out) is used. Model selection is performed by Reversible Jump MCMC. This strategy departs from Park and Casella (2008), who suggest a latent-variable model demanding sampling from each conditional $\beta_i|\beta_j, \forall i \neq j$, since a mixture prior with a point-mass at zero is placed on each β_i. Here, the implementation requires no special prior and retains the joint sampling from the full β vector of non-zero entries.

For k-NN regression, the user must input the number of neighbors considered when predicting the dependent variable, that is, k must be predefined. A low value makes the model more responsive and may improve in-sample goodness-of-fit, but may also lead to overfitting. On the other hand, a high value of k yields smooth predictions, failing to capture the dynamics of the data. In the simulations executed, $k = 25$ performed decently in terms of out-of-sample accuracy.

In the case of Random Forests, the hyperparameters have been calibrated empirically. The number of trees in the forest was set to 20, without a maximum depth. The minimum number of samples required to split an internal node is 4, and the chosen training criterion to measure the quality of a split was the MSE. The minimum number of training samples required in each left and right branches of a split point was set equal to 1. BART has been tuned in the same way, resulting in 200 trees. The prior for the error variance is the inverted chi-squared. Fitting is implemented via MCMC. Both models are trained using a version of the dataset treated using PCA due to the large amount of time series, exponentially increasing the computing time.

Factor models are implemented in line with Bai and Ng (2008) and Medeiros et al. (2019), who propose targeting the predictors. The strategy is to compute the principal components only of those variables with explanatory power when forecasting inflation. Inspired by Medeiros et al. (2019), the algorithm can be summarized as follows:

1. Using lagged values of π_t as controls, regress π_{t+h} on the vector of candidate variables, $\mathbf{X}_t$, and compute the corresponding t-statistics for the coefficients of each variable;

2. For a given significance level, select all variables whose t-statistics are above the critical value;

3. For the set of variables found using the prior steps, estimate the factors $\mathbf{F}_t$ using PCA;

4. Regress π_{t+h} on $\mathbf{f}_{t-j}$, $j = 0, 1, \ldots, 11$, where $\mathbf{f}_t \in \mathbf{F}_t$ are the factors selected using the BIC (Bayesian Information Criterion).

[1] Available at: https://www.rdocumentation.org/packages/monomvn/versions/1.9-13/topics/blasso. Last access: 25 Dec. 2020.

In what concerns the GARCH model, the option was to implement the GARCH(1,1) version, which is frequently used due to its ability to adjust adequately to many time series. For the ARFIMA, the fractional differencing parameter and the number of lags are chosen iteratively seeking to achieve the best goodness-of-fit. Meanwhile, SARIMA was adjusted by first observing that the inflation time series used in the numerical exercises is seasonally adjusted, as explained in the next section. The number of lags was determined via BIC. For the SETAR model, an analogous process was followed, and it was assumed that two regimes were sufficient, for they can be associated with economic cycles (periods of recession and expansion).

In parallel, the VECM model implemented replicates the approach by Cologni and Manera (2008). One distinction is that the authors use quarterly data, while the present work employs monthly time series. Therefore, it was necessary to replace the GDP used by the authors by a correlated monthly variable or a monthly estimate of GDP. Although nowcasts of GDP could be an alternative, industrial production was preferred because it is readily available in the dataset considered here and it is highly correlated with GDP due to the composition of the later. So as to determine the number of lags, several combinations were tested until the one with highest quality, measured in terms of out-of-sample performance, was obtained.

With respect to the moving average model, the reference is Atkeson and Ohanian (2001), who evaluate whether the conventional wisdom that modern Phillips curve-based models are appropriate for inflation forecasting. The authors compares the performance against a naive model assuming that expected inflation over the next four quarters is equal to the inflation over the previous four quarters. That is:

$$E_t(\pi_{t+4} - \pi_t) = 0 \tag{4.1}$$

Here, π_t is the percentage change in the inflation rate between quarters $t-4$ and t. The conclusion is that, for the period considered, no version of the Phillips curve makes more accurate inflation forecasts than those from the naive model. Therefore, including this approach as a benchmark is suitable and advisable.

The Bayesian regression is the Bayesian version of the conventional OLS for a linear regression. It is a traditional method in Statistics, explaining why a full coverage is deemed dispensable; details can be found in Gelman et al. (2004), among others. Here, we replicate the classical assumptions based on a Gamma-Gaussian conjugate prior. Estimation is fulfilled via Evidence Maximization. In a similar manner, Gradient Boosting is implemented with a quadratic loss to resemble OLS for linear regression.

Regarding the multilayer perceptron, a network fully connected with 6 hidden layers and 200 units per layer was designed and fitted. As expected, increasing the number of layers yielded positive results, but a deeper network than the one considered did not improve the out-of-sample performance substantially. The amount of units per layer has also been tuned empirically, aiming to optimize the accuracy when evaluating the test sample.

Finally, the standard LSTM is formed by two sequential LSTM layers constituted by 50 cells. The network was trained using the optimization algorithm NADAM with a maximum of 70 epochs and a batch size of 32. The reasoning behind the setting of these particular parameters is scrutinized in section 4.5.

4.2 Data

The database employed to fit the model developed in this work and the selected benchmarks was provided by McCracken and Ng (2016). The authors maintain in their website[2] a public macroeconomic database comprising 134 monthly US time series freely available at FRED (Federal Reserve Economic Data, which belongs to the Federal Reserve Bank of St. Louis). These time series had to be supplemented by the PMI (Purchasing Managers' Index) compiled by the Institute for Supply Management (ISM), which, at the time of this publication, is no longer available at FRED. The variables are grouped in the following categories (a full description of the dataset is provided in Appendix A):

1. Output and income (17 time series);
2. Labor market (32 time series);
3. Housing (10 time series);
4. Consumption, orders, and inventories (13 time series);
5. Money and credit (14 time series);
6. Interest and exchange rates (22 time series);
7. Prices (21 time series); and
8. Stock market (5 time series).

The selection of this database for model estimation is justified for multiple reasons. Primordially, McCracken and Ng (2016) implemented the best practices reported in the literature to design a database convenient for empirical analysis that requires big data. Hence, it seems appropriate for the intent of this work. Furthermore, this data has been extensively used in the literature in similar studies; for example, see Medeiros et al. (2019). Finally, the time series provided are lengthy, covering several decades and economic cycles.

For the purposes of this work, the period analyzed ranges from January 1978 to December 2019, amounting to 504 observations, which is a size deemed significantly

[2] https://research.stlouisfed.org/econ/mccracken/fred-databases/. Last access: 28 Dec. 2020.

superior to the number of variables contained in the dataset and is sufficient for model estimation with controlled errors. Inflation is gauged by US Consumer Price Index (CPI) for All Urban Consumers (code “CPIAUSCL” in the FRED database), which is a measure of the average monthly change in the price for goods and services paid by urban consumers in United States between any two periods.

In terms of transformations to which each series have been subject, details are available in Appendix A. In addition to those operations, all the time series have been normalized prior to being used to fit the ConvLSTM model and its benchmarks. Whenever required, series are seasonally adjusted, and this is the case of the CPI.

4.3 Training, Validation, and Test Sets

As highlighted by Goodfellow, Bengio and Courville (2016), the central challenge in machine learning is that models must perform reasonably well on new, previously unseen inputs. That is, they must generalize the knowledge acquired from the sample of observations provided, apply it to unobserved inputs, and produce satisfactory out-of-sample performance. In that sense, there is a fundamental difference between machine learning and optimization: not only the training error must be low, but also the test error, since the later is directly related to the degree of generalization.

In this scenario, a popular solution is to split the input data into training and test samples. It is also usual to separate data for validation during the training stage. Consequently, the dataset described in section 4.2 was split as follows (see Figure 19). First, 20% of the observations were reserved for testing. Of the remaining data, 90% of the observations, or 72% of the complete dataset, were applied for training and the rest, for validation. These percentages are aligned with guidelines available in the literature.

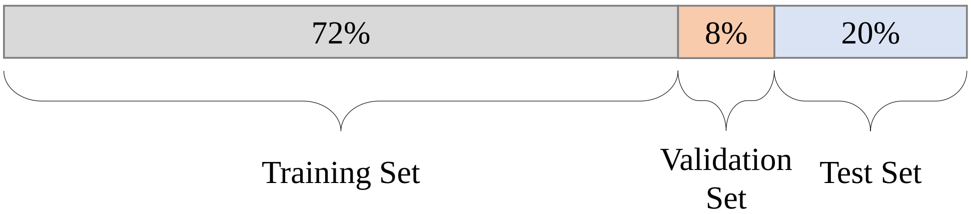


Figure 19 – Simplified view of the splitting process of the dataset used to train and test the model.

When dealing with time series, the preceding splitting process must be adapted. Indeed, random splits will certainly disrupt the autocorrelation of the series, making them improper for fitting any model. An alternative is to implement this procedure in blocks or, equivalently, including the lags of the time series as new variables. This way, a split will not mask the autocorrelations, for every observation of a given variable in a particular instant will be accompanied by the respective observations of that same variable, but shifted backwards in time.

Besides the splitting process detailed, strategies for improving the reliability of the out-of-sample performance analysis have been considered. One of the favored approach is k-Fold Cross-Validation, as reported in James et al. (2013) and Kuhn and Johnson (2013). It is a widespread method due to its simplicity and efficiency. Entails randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fitted on the remaining $(k - 1)$ folds (JAMES et al., 2013). The general procedure works as follows:

1. Initially, the dataset is shuffled randomly;

2. The dataset is then split into k equally sized subsets;

3. For each possible combination available, the following steps are carried out: (a) a single subset is selected as a test sample, while the remaining are employed as a training sample, a share of which is also used for validation; (b) the model is fit on the training set and evaluated on the test set; and (c) evaluation metrics are computed using the test sample so as to assess the out-of-sample performance;

4. Finally, the skill of the model is summarized using the sample of model evaluation metrics.

The value of k must be selected carefully, since there is a bias-variance trade-off involved which may compromise the assessment of the model skill. To summarize the dilemma, as discussed by Kuhn and Johnson (2013), the trade-off is associated with the choice of k in k-fold cross-validation. Typically, given these considerations, one performs k-fold cross-validation using $k = 5$ or $k = 10$, as these values have been shows empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance. It should be noted that, as k gets larger, the difference in size between training set and the resampling subsets gets smaller. As this difference decreases, the bias of the technique becomes smaller. For this study, $k = 10$ has been adopted.

Furthermore, Monte Carlo simulation supplemented cross-validation so as to enhance the reliability of the empirical analysis. More specifically, the splitting of the dataset into training, validation, and test sets has been repeated 100 times, yielding different splits. This strategy was deemed important because, despite the power of cross-validation, the test set remains unchanged during the process. Simulation also increased the number of times the out-of-sample performance of each model was measured, allowing the computation of confidence intervals for these metrics, as explained subsequently in chapter 5.

Finally, it should be noted that the approach based on splitting the input data into different samples imposes some daunting challenges to the implementation of traditional time series models, such as ARIMA or GARCH. In order to circumvent this hindrance, bootstrapping was elected. Concisely, the technique estimates the model coefficients via an initial training set and simulates the fitted model using Monte Carlo to generate new samples, which are separated into new training, validation, and test samples.

4.4 Programming Languages

The models discussed herein were implemented mainly using Python (version 3.7.9). R (version 4.0.2) was resorted to whenever a function could not be readily found in Python or a more efficient routine existed in R. All functions are CPU-based, that is, parallel GPU processing was not considered here to reduce execution time, but this possibility could be exploited in future studies. The routines have been executed in a PC with a Intel Core i7-7700HQ running Microsoft Windows with 16 GB of RAM.

For machine learning models, Tensorflow (version 2.3.0), Keras (version 2.4.3), and Scikit-Learn (version 0.23.2), also known as SKLearn, were used. The former is an end-to-end open source platform for machine learning. Keras is built on top of Tensorflow and offers a consistent and user-friendly API for the implementation of a wide selection of deep learning architectures. SKLearn provides several regression models elected as benchmarks, such as Ridge regression, LASSO and Random Forests. Other libraries accessed include:

- Monomvn (version 1.9-13);
- Glmnet (version 4.0-2);
- BayesTree (version 0.3-1.4);
- Rugarch (version 1.4-4);
- TsDyn (version 10-1.2); and
- Forecast (version 8.13).

The complete list of models, languages, libraries, and functions resorted to in the computational implementation is compiled in Table 2. All the routines implemented by the author of this dissertation, in their full version, are found in the following Github page: https://github.com/AlexandreFT31/Machine_Learning_Inflation.

4.5 Training and Optimization of Neural Networks

A fundamental aspect of neural networks relates to the choice of loss function and optimization algorithm. Many eligible options are available in the literature, and a exhaustive review is outside the scope of this work. Still, due to the importance for

Table 2 – Programming languages, libraries, and main functions used in the implementation of the models discussed in the present dissertation.

Model	Language	Library	Functions
LSTM	Python	Tensorflow and Keras	LSTM
ConvLSTM	Python	Tensorflow and Keras	ConvLSTM2D
Autoencoder	Python	Tensorflow and Keras	Dense, Lambda
MLP	Python	SKLearn	MLPRegressor
Ridge	Python	SKLearn	Ridge, RidgeCV
Bayesian Ridge	Python	SKLearn	BayesianRidge
LASSO	Python	SKLearn	Lasso, LassoCV
Bayesian LASSO	R	Monomvn	Blasso
Elastic Net	R	Glmnet	Glmnet
SVR	Python	SKLearn	Make_pipeline
Random Forest	Python	SKLearn	RandomForestRegressor
KNN Regression	Python	SKLearn	KNeighborsRegressor
BART	R	BayesTree	Bart
Bagging	Python	SKLearn	BaggingRegressor
Huber Regression	Python	SKLearn	HuberRegressor
Theil-Sen Regression	Python	SKLearn	TheilSenRegressor
Factor Regression	Python	SKLearn	PCA, OLS
GARCH	R	Rugarch	UGARCHspec
VECM	R	tsDyn	VECM
SETAR	R	tsDyn	SETAR
SARIMA	R	Forecast	Arima
ARFIMA	R	Forecast	Arfima
GradBoost	Python	SKLearn	GradBoostRegressor
AdaBoost	Python	SKLearn	AdaBoostRegressor
Bayes Regression	Python	SKLearn	ARDRegressor

out-of-sample performance, a review, even a concise one, is called for to justify the design of the model proposed for inflation forecasting.

With respect to loss functions, the most frequent in the literature is the mean squared error (MSE). For instance, Chong, Han and Park (2017) survey and compile several papers and a significant share use MSE and/or a related measure, RMSE (Root Mean Squared Error). This preference is explained by the fact that, in many regression problems, satisfactory results are delivered. Besides, this function is also usual when fitting many time series models, making the comparison between machine learning and traditional econometric models easier. Notwithstanding, there are alternatives. For instance, mean absolute error (MAE) is an option fairly robust to outliers. Analogously, Huber loss, defined by:

$$L(\varepsilon) = \begin{cases} \frac{1}{2}\varepsilon^2, & \text{if } |\varepsilon| \leq \delta \\ \delta\left(|\varepsilon| - \frac{1}{2}\delta\right) & \text{if } |\varepsilon| > \delta \end{cases} \tag{4.2}$$

where ε is the measured error, also handles outliers by placing higher loss on absolute errors above a certain threshold δ. Among these and other possibilities, the decision was to proceed with MSE due to the strong results obtained. Evidently, a detailed examination of this modeling aspect could lead to improvements, but it is outside of the scope of this work and, therefore, is left for future extensions.

Regarding the selection of the optimization algorithm for training, unfortunately no consensus is available. However, a recent and promising approach was advanced by Kingma and Ba (2015), called ADAM ("ADAptive Moment estimation"), which is an algorithm for first-order gradient-based optimization method of stochastic objective functions. ADAM behaves well for non-convex objective functions, which are pervasive in machine learning. In an empirical analysis conducted by Soydaner (2020), even though there is some heterogeneity in the relative performance against other optimization algorithms, ADAM displayed satisfactory results in multiple applications, achieving low mean-squared errors in decent computing time.

Although the full outputs are not reported here and a detailed analysis is not the purpose of the present work, ADAM, together with Nesterov momentum and a quadratic loss function, has also generated the best performance out-of-sample in the context of inflation forecasting, justifying their selection when fitting the neural networks models considered here. Nesterov momentum accelerates the convergence of gradient-based algorithms, explaining its inclusion; see Dozat (2016) for further details.

Finally, it is appropriate to discuss the influence of the number of epochs and the batch size in the optimization process. By definition, each *epoch* corresponds to a complete pass through the training dataset, differing from the *batch size*, which determines the number of samples processed prior to updating the model parameters via the learning (optimization) algorithm. Unfortunately, no consensus exists in the literature regarding adequate values for these hyperparameters. They must be calibrated in an *ad hoc* way, usually choosing as criteria the performance derived out-of-sample, i.e. the generalization power of the network.

Naturally, there is a trade-off between in-sample and out-of-sample performance when setting the number of epochs. As the number increases, the network will pass through the training dataset more times, having greater opportunity to learn the features of the inputs and, thus, improving the prediction accuracy in-sample. However, at the same time, the likelihood of overfitting also increases, for the network may learn how to accurately reproduce the training dataset and, in the process, it may lose generalization power because its coefficients have been excessively tuned to minimize the loss function in the training dataset.

Therefore, the loss function out-of-sample starts decreasing as the number of epochs increase (underfitting region), achieves a minimum at a certain point and, subsequently, starts increasing due to overfitting. This can be seen in Figure 20, where the MSE out-of-sample attains a minimum value for a number of epochs between 35 and 40. For this reason, a total number of 70 epochs seemed reasonable to train every neural network model considered in this work, find the minimum of the out-of-sample loss function, and select the corresponding parameters.

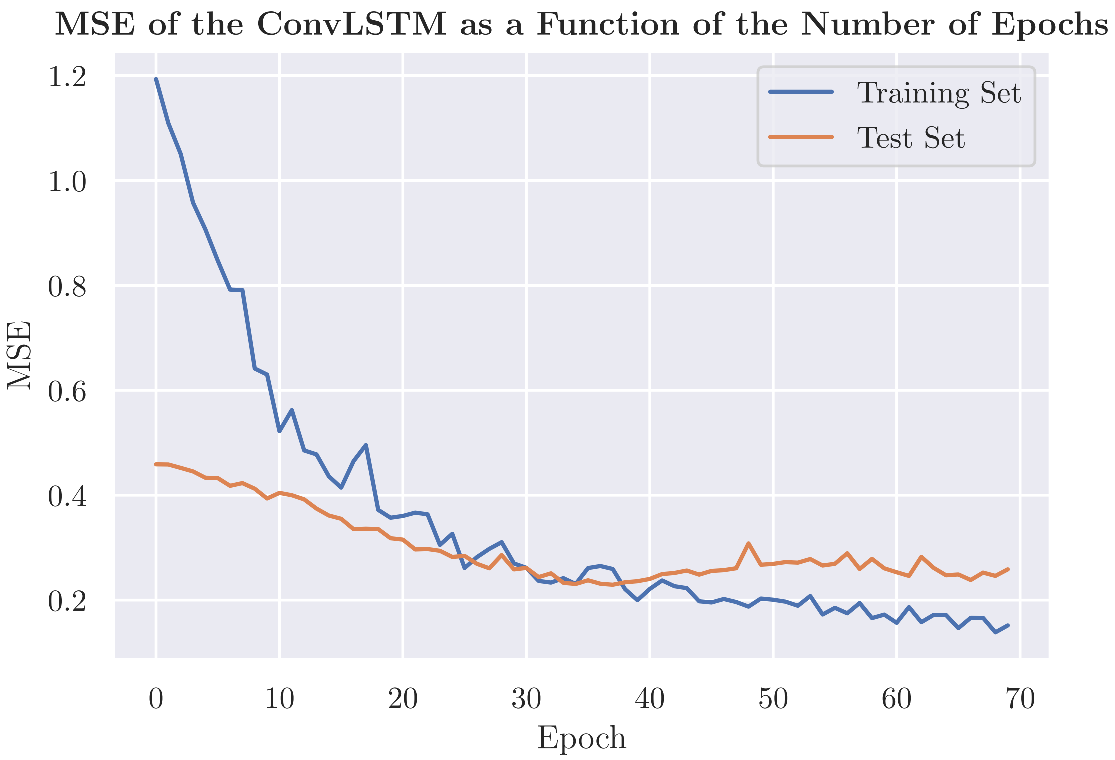


Figure 20 – Illustration of the behavior of the MSE of the ConvLSTM model proposed in this work for inflation forecasting. The batch size is equal to 32. The test MSE begins decreasing until it reaches a minimum around 35-40 epochs. After that point, it increases, despite the fact that the training MSE keeps declining. Consequently, there are no gains in terms of generalization in training the model afterwards.

The effects of the batch size on the learning process are more complex and technical. According to Keskar et al. (2017), in practice, for deep networks, it has been observed that large batches lead to a stark degradation in the quality of the model, as measured by its generalization power. The authors also claim that empirical results show that the lack of generalization arises from the fact that large-batch methods are inclined to converge to *sharp minimizers* of the loss function f, which are characterized by large positive eigenvalues of $\nabla^2 f(x)$ (i.e. strong positive curvature), as illustrated in Figure 21. The large sensitivity of f in the neighborhood of a sharp minimizer negatively impacts the ability to generalize on new data.

On the other hand, small batches often yield *flat minimizers*, marked by small positive eigenvalues of $\nabla^2 f(x)$, leading to enhanced out-of-sample performance (KESKAR et al., 2017). The downside of using a limited batch size is that the convergence to a global optimum may not occur, and the path towards convergence to a local minimum may be noisy due to the constant update of the gradient with few observations, compromising

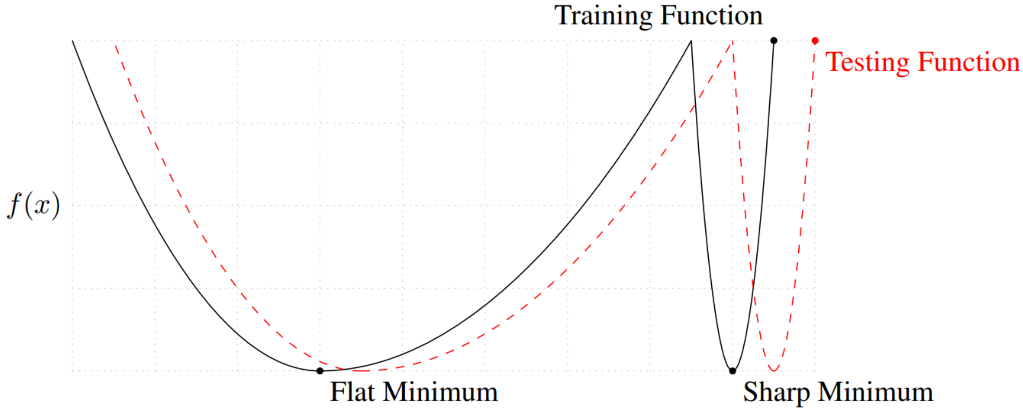


Figure 21 – Conceptual sketch of flat and sharp minima. The loss function is denoted by $f(x)$. Observe that using the estimated sharp minimum of the training function leads to a substantially higher value of f using the test set. In contrast, the difference between the training and test values are much more modest for a flat minimum. Source: Keskar et al. (2017).

accuracy, as argued by Goodfellow, Bengio and Courville (2016). It is worth stressing that the minimization of the loss functions of neural networks is typically a non-convex optimization problem, meaning that multiple local minima usually exist and the path towards the global minimum may be nontrivial to locate.

Hence, finding an equilibrium between these objectives (generalization vs. smooth, fast convergence) by specifying an appropriate batch size is crucial, because it guarantees convergence in a decent speed while benefiting from the noise to escape from basins of attraction of sharp minima towards flatter minima that have better generalization power, as argued by Hoffer, Hubara and Soudry (2017). In addition, the noise added by small batches can promote a regularizing effect.

In the simulations carried out in this work, the batch size was defaulted to 32, a value in the range of 32 to 512 frequently considered in practice to balance the trade-off between convergence and generalization power (KESKAR et al., 2017). It should be noted that batch sizes are usually defined as powers of 2 to offer better run time, especially when using GPUs (GOODFELLOW; BENGIO; COURVILLE, 2016). Thus, the next value after 32 would be 64, which is regarded as too large for the dataset of 504 observations considered in this work.

Indeed, in the experiments conducted by Keskar et al. (2017), large batches were constructed using 10% of the training data. Here, this dataset (including the share used for validation) is formed by 403 observations, implying that a batch size of 64 would correspond to approximately 16% of the training data and, thus, would be far too large. In fact, even a batch size of 32 is somewhat large, as indicated by this analysis. However, a smaller batch would render the gradient estimate too unstable and inaccurate, damaging the learning ability of the network. For the sake of completeness, although not reported here, other sizes were tested, yielding poor outcomes with respect to a batch size of 32.

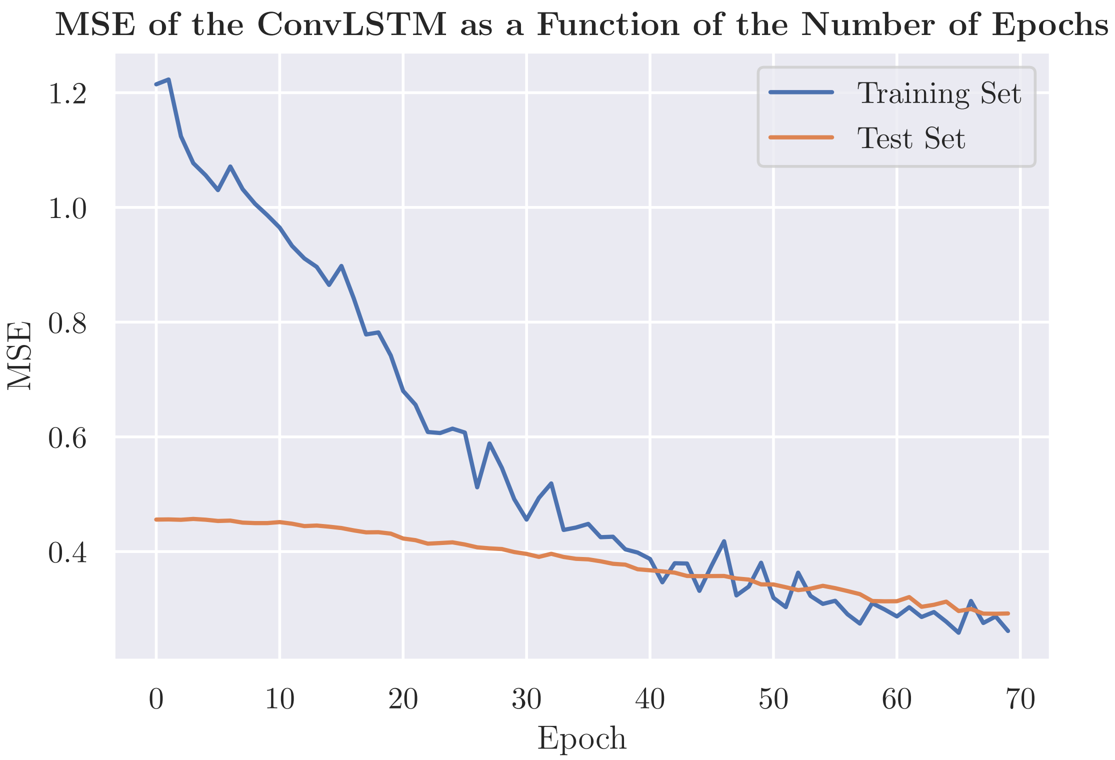


Figure 22 – Evolution of the MSE of the ConvLSTM trained in Figure 20, but now trained with a batch size of 256. The MSE curve is naturally smoother, since the estimates of the gradient are more precise. However, due to the deteriorated generalization power, even after 70 epochs, the out-of-sample MSE is still above the minimum reached when the network was trained with a batch size of 32. The same happens with the in-sample values.

5 RESULTS

In this section, the results of the analysis of the out-of-sample relative performance of the model proposed for inflation forecasting are presented. Initially, we analyze the US CPI time series, identifying stylized facts. Later, the performance metrics are introduced and a method for building confidence intervals is discussed. Next, the main results are shown and debated.

5.1 Stylized Facts in Inflation Time Series

As a background to the discussion of the performance of the models tested, we describe the empirical properties of the inflation time series. The first difference of the log values is exhibited in Figure 23. The profile suggests that, after the transformation, no persistence remains and, thus, the series does not have unit roots, which can be confirmed via the Augmented Dickey-Fuller (ADF) test. Also, inflation seems nonstationary due to time-varying volatility, which is shown in Figure 24.

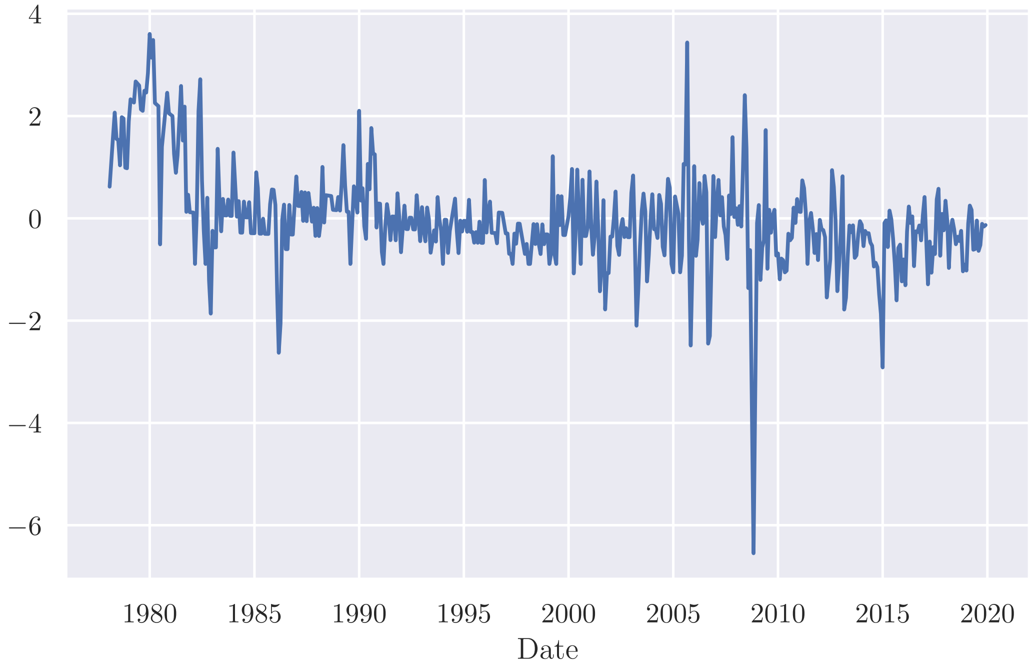


Figure 23 – Normalized first difference of the log prices, as measured by the US CPI (CPIAUCSL series in the FRED database).

Moreover, as the Q-Q plot in Figure 25 shows, the hypothesis of normality may be immediately rejected, and, even though the results are not reported here, this conclusion can be confirmed by a simple Jarque-Bera test. Indeed, extreme observations have been reported far more often than implied by the normal distribution, signaling that the true

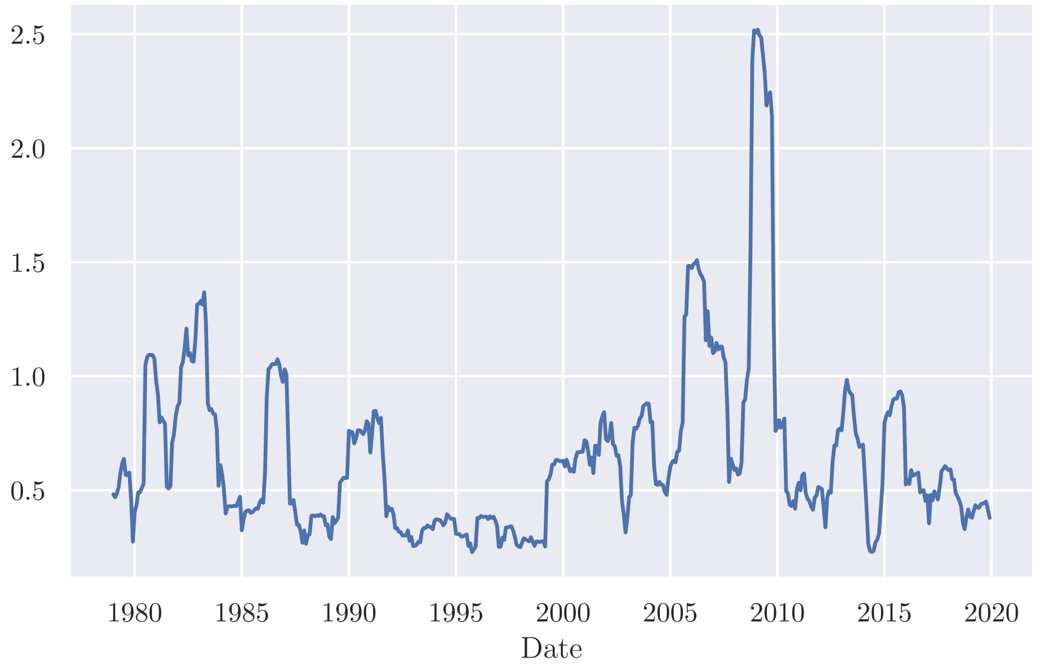


Figure 24 – Standard deviation of the first difference of the log prices, as measured by the US CPI (CPIAUCSL series in the FRED database). Computed using a 12-month rolling window.

distribution must have positive excess kurtosis. Such conclusion is consistent with Monache and Petrella (2017). These observation have typically occurred during turbulent periods, such as the late 70s, when the US economy suffered from high and uncontrollable inflation, and the 2008-09 financial crisis.

The autocorrelation and partial autocorrelation functions of the first difference of the log inflation are plotted in Figure 26 and Figure 27. Both plots show that lags have predictive power to explain current inflation. There are no apparent evidence of long-term memory in the series, despite the fact that the autocorrelation function decays somewhat slowly and several lags are statistically significant.

Since McCracken and Ng (2016) advise in favor of the use of the second difference of the log prices, the respective autocorrelation and partial autocorrelation functions are also displayed in Figure 28 and Figure 29. This additional differencing corrects the autocorrelation function, which now decays exponentially, leaving few statistically significant lags. The partial autocorrelation function remains well-behaved.

5.2 Performance Metrics

The criteria adopted to compare the performance of each model is comprised by five metrics computed for the set of out-of-sample predictions. These metrics are well-known in

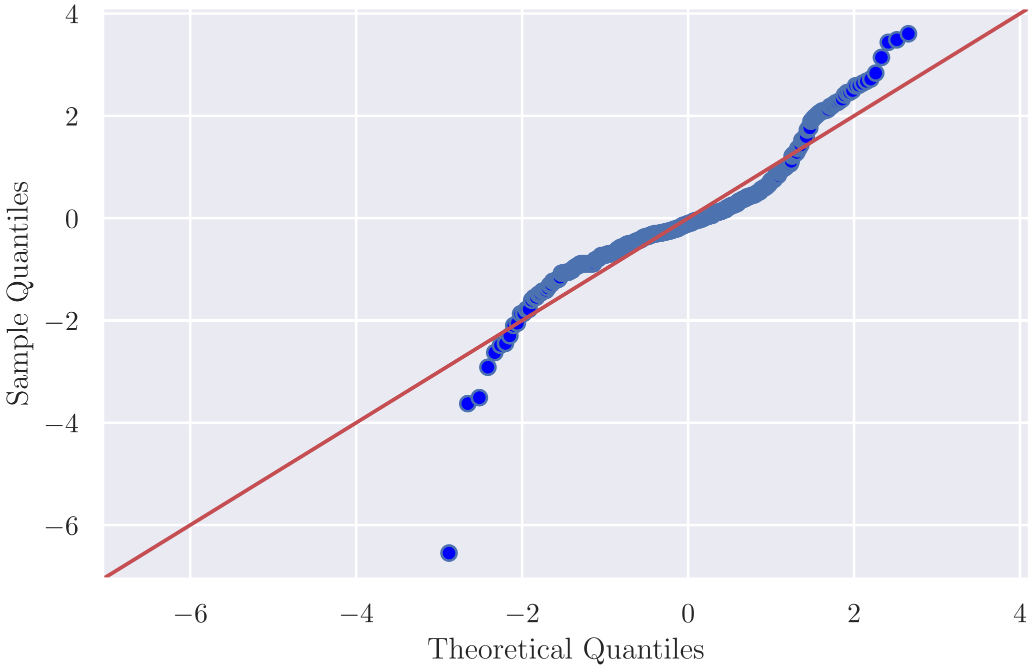


Figure 25 – Q-Q plot of the normalized first difference of the log prices, as measured by the US CPI (CPIAUCSL series in the FRED database).

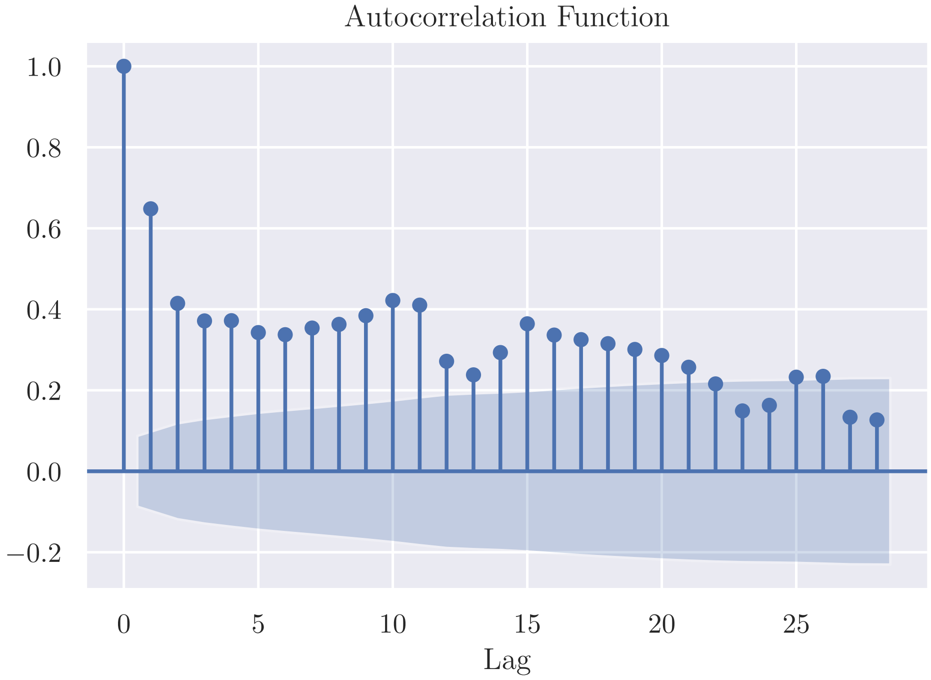


Figure 26 – Autocorrelation function of the first difference of the log prices, as measured by the US CPI. Confidence interval (shaded area) is computed using Bartlett's formula with a significance level of 5%.

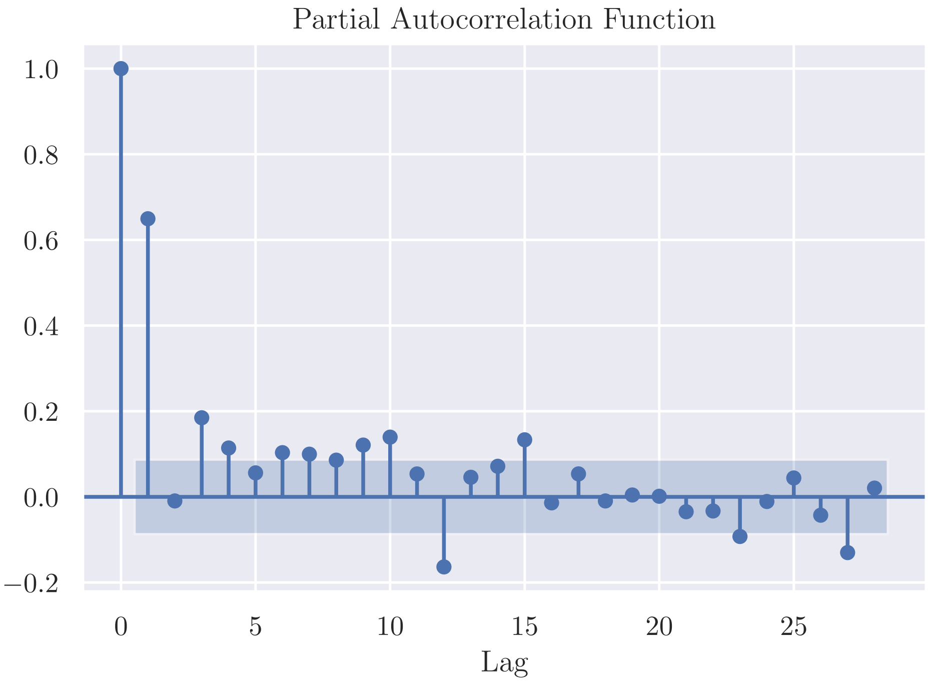


Figure 27 – Partial autocorrelation function of the first difference of the log prices, as measured by the US CPI. Confidence interval (shaded area) is computed using Bartlett's formula with a significance level of 5%.

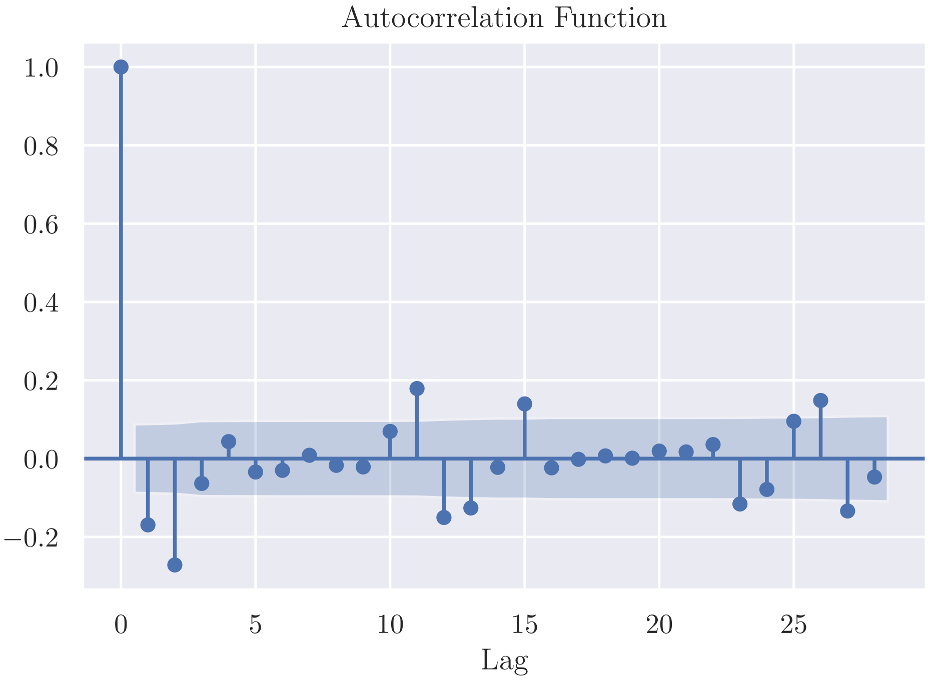


Figure 28 – Autocorrelation function of the log inflation series after twice-differencing. Confidence interval (shaded area) is computed using Bartlett's formula with a significance level of 5%.

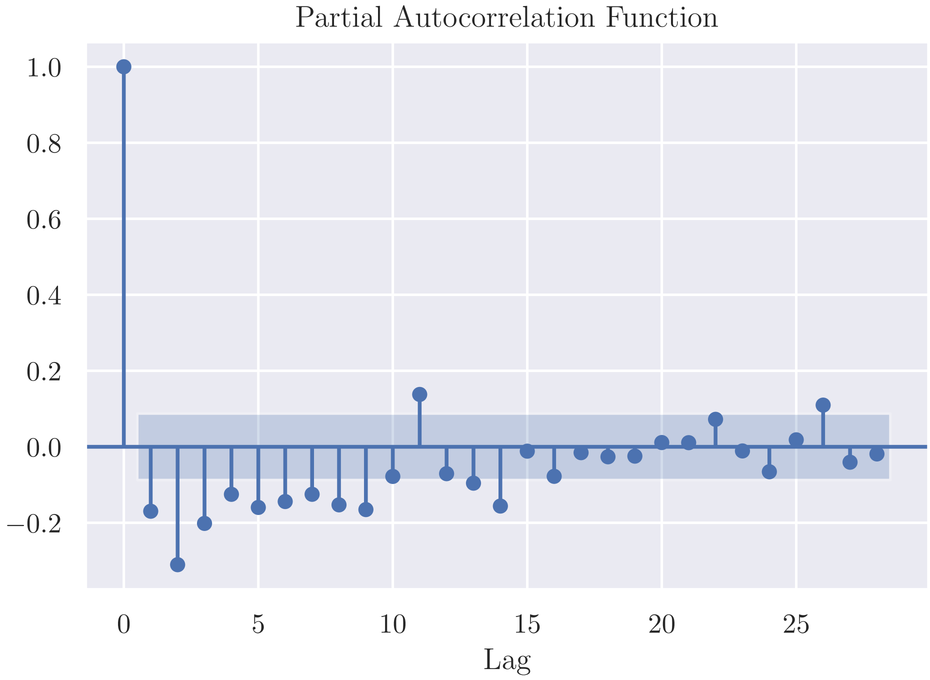


Figure 29 – Partial autocorrelation function of the log inflation series after twice-differencing. Confidence interval (shaded area) is computed using Bartlett's formula with a significance level of 5%.

the machine learning literature and many empirical studies resort to them as a means to compare different models; for instance, see Atsalakis and Valavanis (2009) and Chong, Han and Park (2017). In the next expressions, n is the number of predictions, y_i, $i = 1, 2, \ldots, n$, are the observed values, and $\hat{y}_i$ are the forecast values.

1. Mean squared error (MSE): corresponds to the average of the squared prediction errors.
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.1}$$

2. Mean absolute error (MAE): equals the average absolute prediction error.
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5.2}$$

3. Root mean squared error: it is simply the square root of the MSE.
$$RMSE = \sqrt{MSE} \tag{5.3}$$

4. Mean absolute percentage error (MAPE): it is equivalent to the average absolute relative prediction error.
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{5.4}$$

5. Cosine similarity (CS): is a measure of similarity between any two non-zero vectors based on the canonical inner product of $\mathbb{R}^2$. Mathematically, corresponds to the cosine of the angle between these two vectors. Hence, it lies in the range $[-1, 1]$ and is analogous to the concept of correlation.

$$CS = \frac{\sum_{i=1}^{n} y_i \hat{y}_i}{\sqrt{\left(\sum_{i=1}^{n} y_i^2\right)\left(\sum_{i=1}^{n} \hat{y}_i^2\right)}} \tag{5.5}$$

5.3 Confidence Intervals and Hypothesis Testing

In line with chapter 4, where data and methodology are debated, the use of cross-validation and Monte Carlo simulation permitted gauging multiple times the out-of-sample performance of the proposed model and its benchmarks. In that sense, 100 simulations, together with a 10-fold cross-validation, conceived 1.000 measurements, with which confidence intervals and hypothesis testing could be conducted.

Intrinsically, the theoretical distribution of the performance metrics listed in the previous subsection, although unknown, most certainly depart from the normal distribution. One reason for this deviation is that these metrics are truncated. For example, MSE and MAE are always greater than 0. Hence, the conventional formulas for confidence intervals and hypothesis testing for normally distributed variables cannot be reliably employed here.

A method to effectively overcome the aforementioned challenge entails *kernel density estimation* (KDE). Briefly, KDE is a nonparametric method to estimate the probability distribution function of a random variable from which a sample of realizations is available. Let $(x_1, \ldots, x_n)$ be a i.i.d. sample of size n drawn from a random variable with unknown density f. The *kernel density estimator* of f, $\hat{f}(x; h)$, is given by:

$$\hat{f}(x; h) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{5.6}$$

where $K(\cdot)$ is a non-negative function denominated *kernel* and $h > 0$ is the *bandwidth*, which acts as a smoothing parameter. Although other functions exist in the literature, the Gaussian kernel is a popular choice in many cases; see James et al. (2013). The bandwidth h can be calibrated empirically. A rule-of-thumb is provided by Silverman (1986), which is given by:

$$h = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{1/5} \approx 1.06\hat{\sigma}n^{-1/5} \tag{5.7}$$

where $\hat{\sigma}$ is the estimated standard deviation.

Through the estimate produced by the KDE, confidence intervals are built by numerically integrating $\hat{f}(x; h)$ to find the cumulative distribution function. Numerical integration is relatively simple in this context due to the fact that the estimate is a continuous function. Basic algorithms, such as the composite trapezoid rule, perform decently here. It is worth highlighting that, in the analysis discussed in the next subsections, all calculation were carried out using a significance level of 5%.

5.4 Empirical Analysis and Discussion

Initially, the discussion begins by analyzing the individual results exhibited by the ConvLSTM model. Figure 30 and Figure 31 display the empirical probability distribution and the boxplot of the MSE and MAE metrics, respectively. It is possible to see that, across simulations, the performance of the model naturally oscillates, but within a tight range, and never reaching high loss levels. Indeed, as it is revealed in the next tables, the ConvLSTM produces losses constrained to a lower confidence interval than its benchmarks for the most relevant horizons.

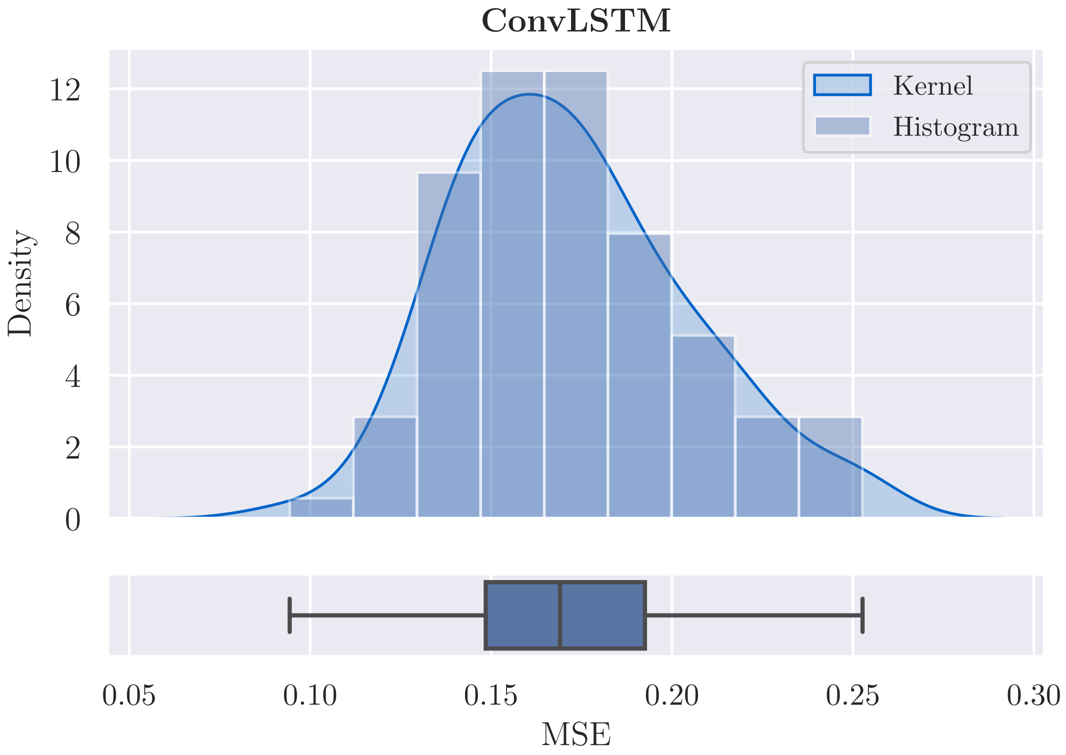


Figure 30 – Histogram (upper panel, normalized) and boxplot (bottom panel) of the MSE of the ConvLSTM model proposed. The kernel density estimation was carried out using a Gaussian kernel.

The next tables contain the out-of-sample performance of each model fitted according to the methodology detailed in chapter 4. Firstly, Table 3 presents the descriptive statistics of the out-of-sample MSE for all forecasting horizons (1, 2, 3, 6, and 12 months ahead) as well as for the cumulative forecasts over 3, 6, and 12 months. The ranks according to the average MSE, the confidence intervals, and the average MSE reduction delivered by each model with respect to the random walk predictions are shown in Table 4, Table 5, and Table 6, respectively.

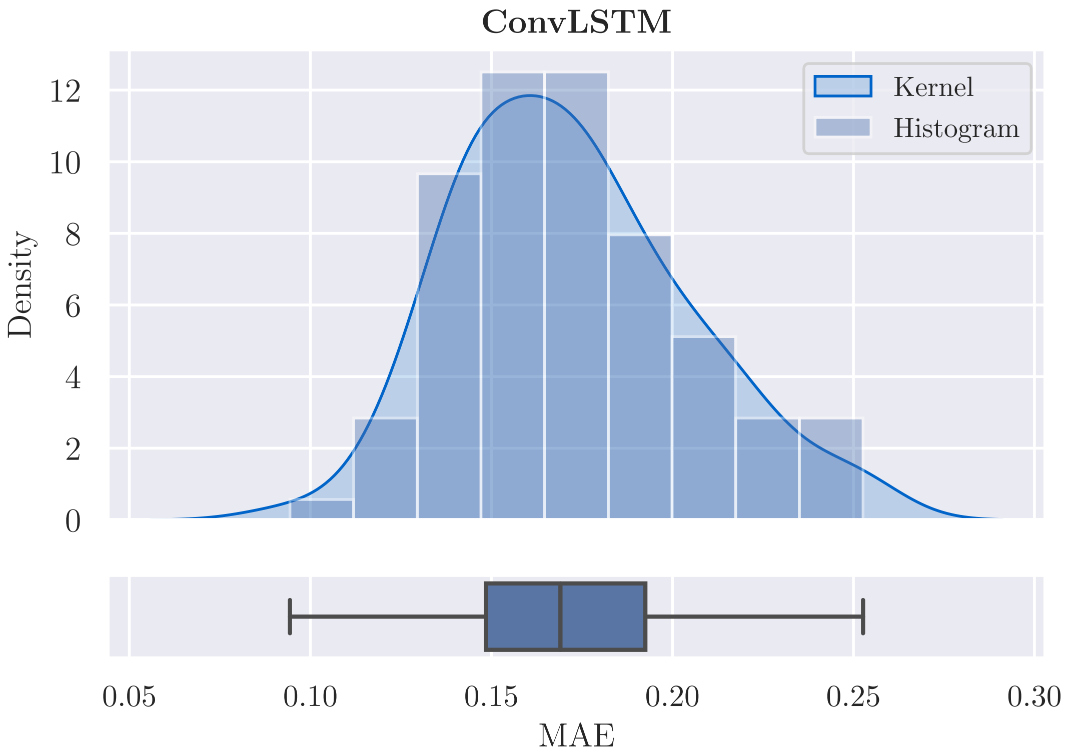


Figure 31 – Histogram (upper panel, normalized) and boxplot (bottom panel) of the MSE of the ConvLSTM model proposed. The kernel density estimation was carried out using a Gaussian kernel.

Table 3 – Descriptive statistics of the out-of-sample MSE of each model for all forecasting horizons (1, 2, 3, 6, and 12 months ahead) as well as for the cumulative forecasts over 3, 6, and 12 months. Throughout this section, the notation used to designate each model works as follows: "MLP" refers to the multi-layer perceptron; "RW" is the random walk; "Ridge CV", "LASSO CV", and "Enet CV" are the Ridge, LASSO, and Elastic Net regressions with parameters chosen via cross-validation; "BRidge" is the Bayesian Ridge; "BLASSO" is the Bayesian LASSO; and "MA" is the moving average model suggested by Atkeson and Ohanian (2001).

Model	**MSE**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
LSTM	Min.	0.14	0.30	0.40	0.37	0.39	0.24	0.28	0.30
	Q1	0.24	0.46	0.50	0.50	0.51	0.37	0.47	0.41
	Median	0.32	0.51	0.55	0.55	0.57	0.42	0.55	0.49
	Q3	0.40	0.60	0.60	0.64	0.62	0.49	0.68	0.57
	Max.	0.76	1.12	0.96	1.48	1.15	0.73	1.21	1.15
ConvLSTM	Min.	0.09	0.31	0.39	0.39	0.43	0.22	0.27	0.18
	Q1	0.15	0.40	0.44	0.45	0.46	0.30	0.39	0.29
	Median	0.17	0.42	0.45	0.46	0.46	0.33	0.43	0.34
	Q3	0.19	0.44	0.46	0.46	0.47	0.37	0.50	0.40

Table 3 – Descriptive statistics of the out-of-sample MSE of each model for all forecasting horizons (1, 2, 3, 6, and 12 months ahead) as well as for the cumulative forecasts over 3, 6, and 12 months. Throughout this section, the notation used to designate each model works as follows: "MLP" refers to the multi-layer perceptron; "RW" is the random walk; "Ridge CV", "LASSO CV", and "Enet CV" are the Ridge, LASSO, and Elastic Net regressions with parameters chosen via cross-validation; "BRidge" is the Bayesian Ridge; "BLASSO" is the Bayesian LASSO; and "MA" is the moving average model suggested by Atkeson and Ohanian (2001).

Model	**MSE**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
	Max.	0.25	0.47	0.47	0.47	0.48	0.59	0.86	0.62
MLP	Min.	0.75	0.86	0.88	0.89	0.93	0.88	0.90	0.99
	Q1	0.92	1.10	1.10	1.10	1.09	1.05	1.09	1.11
	Median	0.99	1.16	1.16	1.16	1.18	1.11	1.15	1.17
	Q3	1.07	1.22	1.21	1.25	1.25	1.17	1.24	1.24
	Max.	1.31	1.36	1.34	1.48	1.48	1.32	1.42	1.43
RW	Min.	1.53	1.52	1.60	1.51	1.38	1.46	1.46	1.48
	Q1	1.89	1.84	1.87	1.88	1.81	1.90	1.94	1.91
	Median	2.04	1.99	1.99	2.05	1.97	2.01	2.06	2.07
	Q3	2.15	2.11	2.16	2.19	2.18	2.14	2.19	2.24
	Max.	2.54	2.55	2.61	2.54	2.66	2.43	2.53	2.62
Ridge CV	Min.	0.66	0.88	0.83	0.80	0.82	0.87	0.95	0.94
	Q1	0.80	0.99	0.99	0.99	0.97	0.97	0.99	0.99
	Median	0.85	1.01	1.02	1.03	1.03	0.99	1.00	1.00
	Q3	0.89	1.02	1.03	1.04	1.07	1.01	1.02	1.01
	Max.	1.30	1.07	1.07	1.11	1.18	1.19	1.20	1.15
BRidge	Min.	0.61	0.94	0.85	0.87	0.85	0.83	0.88	0.94
	Q1	0.78	1.03	1.03	1.03	1.02	0.97	1.00	1.02
	Median	0.85	1.07	1.08	1.08	1.10	1.01	1.04	1.05
	Q3	0.91	1.11	1.11	1.14	1.15	1.07	1.10	1.11
	Max.	1.20	1.26	1.23	1.32	1.36	1.21	1.26	1.26
LASSO CV	Min.	0.68	0.84	0.84	0.86	0.81	0.88	0.95	0.95
	Q1	0.78	1.00	1.00	0.99	0.98	0.96	1.00	1.00
	Median	0.81	1.01	1.01	1.02	1.03	1.00	1.00	1.00
	Q3	0.87	1.01	1.02	1.04	1.07	1.00	1.00	1.00
	Max.	1.08	1.09	1.12	1.15	1.22	1.07	1.08	1.05
BLASSO	Min.	0.68	0.84	0.83	0.79	0.81	0.84	0.94	0.98
	Q1	0.78	1.00	0.99	0.99	0.97	0.96	0.99	0.99
	Median	0.83	1.01	1.01	1.02	1.02	0.98	1.00	1.00
	Q3	0.90	1.01	1.02	1.04	1.07	1.00	1.00	1.00

Table 3 – Descriptive statistics of the out-of-sample MSE of each model for all forecasting horizons (1, 2, 3, 6, and 12 months ahead) as well as for the cumulative forecasts over 3, 6, and 12 months. Throughout this section, the notation used to designate each model works as follows: "MLP" refers to the multi-layer perceptron; "RW" is the random walk; "Ridge CV", "LASSO CV", and "Enet CV" are the Ridge, LASSO, and Elastic Net regressions with parameters chosen via cross-validation; "BRidge" is the Bayesian Ridge; "BLASSO" is the Bayesian LASSO; and "MA" is the moving average model suggested by Atkeson and Ohanian (2001).

Model	**MSE**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
	Max.	1.03	1.03	1.03	1.06	1.13	1.12	1.05	1.04
Enet CV	Min.	0.49	0.84	0.83	0.78	0.80	0.83	0.91	0.91
	Q1	0.71	0.98	0.98	0.98	0.95	0.93	0.96	0.97
	Median	0.76	1.00	1.00	1.01	1.01	0.96	0.98	0.99
	Q3	0.83	1.01	1.01	1.03	1.06	0.98	1.00	1.00
	Max.	0.97	1.01	1.02	1.05	1.13	1.00	1.00	1.00
SVR	Min.	0.83	0.86	0.83	0.81	0.81	0.90	0.89	0.94
	Q1	0.90	1.01	1.01	1.00	1.00	0.97	0.99	1.00
	Median	0.92	1.03	1.03	1.05	1.06	0.99	1.02	1.03
	Q3	0.94	1.06	1.05	1.08	1.10	1.02	1.04	1.06
	Max.	0.99	1.14	1.14	1.16	1.35	1.08	1.17	1.20
RF	Min.	0.75	0.92	0.89	0.87	0.84	0.87	0.94	0.93
	Q1	0.87	1.05	1.04	1.05	1.03	1.01	1.02	1.02
	Median	0.93	1.09	1.09	1.10	1.10	1.05	1.05	1.07
	Q3	1.00	1.13	1.13	1.15	1.17	1.08	1.10	1.10
	Max.	1.21	1.36	1.27	1.30	1.47	1.15	1.24	1.22
BART	Min.	0.73	0.91	0.85	0.86	0.83	0.88	0.92	0.92
	Q1	0.86	1.01	1.02	1.02	1.02	0.96	0.98	1.00
	Median	0.91	1.05	1.06	1.07	1.08	1.00	1.01	1.02
	Q3	0.96	1.08	1.10	1.11	1.13	1.04	1.04	1.04
	Max.	1.23	1.16	1.27	1.27	1.35	1.12	1.14	1.13
Bagging	Min.	0.80	0.97	0.89	0.87	0.94	0.90	0.91	0.95
	Q1	0.94	1.12	1.08	1.08	1.10	1.05	1.05	1.05
	Median	0.99	1.16	1.14	1.15	1.17	1.10	1.09	1.12
	Q3	1.06	1.21	1.21	1.21	1.23	1.15	1.14	1.17
	Max.	1.31	1.42	1.41	1.32	1.54	1.28	1.40	1.27
kNN	Min.	0.95	0.86	0.90	0.85	0.88	0.92	0.92	0.88
	Q1	0.98	1.00	1.00	1.01	1.00	1.00	1.01	1.00
	Median	1.00	1.03	1.04	1.05	1.05	1.02	1.03	1.03
	Q3	1.02	1.07	1.07	1.07	1.11	1.05	1.05	1.06

Table 3 – Descriptive statistics of the out-of-sample MSE of each model for all forecasting horizons (1, 2, 3, 6, and 12 months ahead) as well as for the cumulative forecasts over 3, 6, and 12 months. Throughout this section, the notation used to designate each model works as follows: "MLP" refers to the multi-layer perceptron; "RW" is the random walk; "Ridge CV", "LASSO CV", and "Enet CV" are the Ridge, LASSO, and Elastic Net regressions with parameters chosen via cross-validation; "BRidge" is the Bayesian Ridge; "BLASSO" is the Bayesian LASSO; and "MA" is the moving average model suggested by Atkeson and Ohanian (2001).

Model	**MSE**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
	Max.	1.07	1.14	1.14	1.17	1.26	1.12	1.11	1.14
Huber	Min.	0.67	1.68	1.51	1.54	1.54	1.42	1.38	1.55
	Q1	0.93	1.99	2.03	1.98	2.01	1.67	1.88	1.92
	Median	1.04	2.18	2.25	2.32	2.20	1.85	2.04	2.14
	Q3	1.19	2.42	2.43	2.55	2.52	2.04	2.25	2.44
	Max.	1.49	3.22	3.29	3.28	3.19	2.45	2.93	2.97
Theil-Sen	Min.	0.67	1.77	1.51	1.66	1.59	1.46	1.45	1.62
	Q1	0.94	2.08	2.10	2.14	2.11	1.71	1.93	1.97
	Median	1.04	2.31	2.33	2.44	2.33	1.90	2.09	2.19
	Q3	1.20	2.55	2.58	2.68	2.60	2.11	2.30	2.52
	Max.	1.53	3.44	3.49	3.49	3.36	2.57	2.97	3.02
Factors	Min.	0.74	0.87	0.88	0.87	0.86	0.94	0.93	0.90
	Q1	0.93	1.01	1.01	1.02	1.02	1.01	1.01	1.01
	Median	1.01	1.04	1.05	1.07	1.08	1.06	1.05	1.03
	Q3	1.08	1.08	1.08	1.11	1.13	1.10	1.11	1.08
	Max.	1.36	1.28	1.24	3.69	1.29	1.27	1.42	2.89
GARCH	Min.	0.24	0.37	0.38	0.35	0.39	0.50	0.45	0.39
	Q1	0.49	0.72	0.83	0.79	0.82	0.67	0.60	0.64
	Median	0.64	0.93	1.05	1.11	1.09	0.72	0.70	0.70
	Q3	0.89	1.34	1.52	1.52	1.46	0.82	0.80	0.83
	Max.	2.19	3.09	3.31	3.67	3.93	1.20	1.30	1.38
VECM	Min.	0.38	0.83	0.99	0.84	1.10	0.53	0.49	0.36
	Q1	0.63	1.28	1.45	1.27	1.42	0.78	0.79	0.77
	Median	0.70	1.44	1.63	1.48	1.57	0.91	0.96	1.13
	Q3	0.77	1.65	1.94	1.70	1.82	1.05	1.15	1.38
	Max.	1.06	2.23	2.95	2.43	3.31	1.51	1.59	2.08
SETAR	Min.	0.55	0.71	0.79	0.75	0.65	1.04	0.98	0.67
	Q1	0.80	0.99	1.11	1.04	1.04	1.27	1.33	1.53
	Median	0.93	1.16	1.27	1.20	1.17	1.42	1.55	1.81
	Q3	1.06	1.33	1.48	1.36	1.41	1.59	1.74	2.01

Table 3 – Descriptive statistics of the out-of-sample MSE of each model for all forecasting horizons (1, 2, 3, 6, and 12 months ahead) as well as for the cumulative forecasts over 3, 6, and 12 months. Throughout this section, the notation used to designate each model works as follows: "MLP" refers to the multi-layer perceptron; "RW" is the random walk; "Ridge CV", "LASSO CV", and "Enet CV" are the Ridge, LASSO, and Elastic Net regressions with parameters chosen via cross-validation; "BRidge" is the Bayesian Ridge; "BLASSO" is the Bayesian LASSO; and "MA" is the moving average model suggested by Atkeson and Ohanian (2001).

Model	**MSE**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
	Max.	1.37	1.75	2.10	1.83	2.01	1.96	2.23	2.66
MA	Min.	0.65	0.78	0.76	0.71	0.71	0.78	0.72	0.67
	Q1	0.84	1.01	0.99	0.92	0.92	1.01	0.93	0.88
	Median	0.93	1.13	1.12	1.06	1.07	1.12	1.06	1.02
	Q3	1.03	1.27	1.27	1.22	1.24	1.24	1.20	1.19
	Max.	1.33	1.64	1.65	1.58	1.62	1.60	1.56	1.55
SARIMA	Min.	0.54	0.64	0.75	0.87	0.85	0.49	0.47	0.39
	Q1	0.70	0.93	1.02	1.09	1.10	0.67	0.60	0.61
	Median	0.77	1.02	1.13	1.24	1.22	0.75	0.69	0.68
	Q3	0.82	1.14	1.28	1.35	1.34	0.81	0.78	0.77
	Max.	1.05	1.54	1.61	1.82	1.82	1.06	1.05	1.22
ARFIMA	Min.	0.51	0.62	0.88	0.82	0.82	0.45	0.46	0.44
	Q1	0.69	0.83	1.02	1.08	1.08	0.62	0.63	0.62
	Median	0.75	0.95	1.14	1.19	1.21	0.70	0.71	0.71
	Q3	0.82	1.11	1.38	1.34	1.35	0.79	0.81	0.81
	Max.	1.12	1.43	1.76	1.92	1.83	1.09	1.00	1.03
GradBoost	Min.	0.75	0.92	0.92	0.92	0.90	0.91	0.90	0.97
	Q1	0.84	1.07	1.09	1.09	1.07	1.03	1.03	1.05
	Median	0.92	1.13	1.16	1.14	1.15	1.07	1.08	1.10
	Q3	0.98	1.19	1.20	1.20	1.23	1.12	1.13	1.15
	Max.	1.13	1.39	1.35	1.37	1.45	1.27	1.27	1.37
AdaBoost	Min.	0.80	0.91	0.86	0.84	0.80	0.90	0.96	0.94
	Q1	0.88	1.02	1.03	1.03	1.02	0.99	1.02	1.02
	Median	0.93	1.06	1.06	1.07	1.10	1.04	1.06	1.06
	Q3	0.97	1.09	1.10	1.10	1.15	1.08	1.09	1.11
	Max.	1.10	1.22	1.22	1.24	1.38	1.19	1.30	1.26
Bayes Reg.	Min.	0.76	0.96	1.01	1.00	0.98	0.97	0.99	1.00
	Q1	0.99	1.22	1.22	1.23	1.24	1.19	1.20	1.23
	Median	1.07	1.33	1.32	1.33	1.34	1.30	1.31	1.33
	Q3	1.20	1.43	1.42	1.42	1.50	1.40	1.42	1.46

Table 3 – Descriptive statistics of the out-of-sample MSE of each model for all forecasting horizons (1, 2, 3, 6, and 12 months ahead) as well as for the cumulative forecasts over 3, 6, and 12 months. Throughout this section, the notation used to designate each model works as follows: "MLP" refers to the multi-layer perceptron; "RW" is the random walk; "Ridge CV", "LASSO CV", and "Enet CV" are the Ridge, LASSO, and Elastic Net regressions with parameters chosen via cross-validation; "BRidge" is the Bayesian Ridge; "BLASSO" is the Bayesian LASSO; and "MA" is the moving average model suggested by Atkeson and Ohanian (2001).

Model	**MSE**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
	Max.	1.56	1.74	1.95	2.08	2.10	1.90	1.71	1.75

Table 4 – Ranks produced by comparing the models according to their average MSE through the simulations. The two best models are highlight in bold font.

Model	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
LSTM	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**
ConvLSTM	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
MLP	19	19	18	17	17	20	21	21
RW	26	24	24	24	24	26	24	24
Ridge CV	11	7	6	6	6	11	10	9
BRidge	10	14	12	13	12	13	14	14
LASSO CV	8	6	5	5	5	9	9	8
BLASSO	9	5	4	4	4	8	8	7
Enet CV	7	4	3	3	3	7	6	6
SVR	14	8	7	7	7	10	12	13
RF	18	16	13	14	14	16	17	15
BART	12	11	10	10	10	12	11	11
Bagging	20	20	16	16	16	19	20	20
kNN	21	9	8	8	8	14	13	12
Huber	23	25	25	25	25	24	25	25
Theil-Sen	24	26	26	26	26	25	26	26
Factors	22	12	9	11	11	17	18	16
GARCH	4	15	19	19	18	5	4	5
VECM	3	23	23	23	23	6	7	18
SETAR	17	21	21	20	19	23	23	23
MA	15	17	14	12	9	21	15	10
SARIMA	6	10	17	18	21	4	3	3
ARFIMA	5	3	20	21	20	3	5	4
GradBoost	13	18	15	15	15	18	19	19
AdaBoost	16	13	11	9	13	15	16	17

Table 4 – Ranks produced by comparing the models according to their average MSE through the simulations. The two best models are highlight in bold font.

Model	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
Bayes Reg.	25	22	22	22	22	22	22	22

Table 5 – Confidence intervals for the MSE. For each model, the first line contains the average MSE and the second one, the limits of the interval. Some forecasting periods were omitted for the sake of simplicity. The results for these additional periods are similar to those reported in the table.

Model	**1**	**3**	**3M**	**6M**	**12M**
LSTM	0.33	0.55	0.44	0.59	0.52
	(0.13, 0.64)	(0.40, 0.77)	(0.27, 0.69)	(0.36, 1.01)	(0.31, 0.89)
ConvLSTM	0.17	0.44	0.33	0.46	0.36
	(0.12, 0.25)	(0.39, 0.47)	(0.23, 0.47)	(0.29, 0.73)	(0.21, 0.59)
MLP	1.00	1.16	1.11	1.16	1.18
	(0.81, 1.24)	(0.93, 1.33)	(0.93, 1.30)	(0.99, 1.38)	(1.02, 1.37)
RW	2.02	2.01	2.02	2.06	2.08
	(1.61, 2.45)	(1.63, 2.46)	(1.69, 2.40)	(1.62, 2.46)	(1.65, 2.54)
Ridge CV	0.86	1.01	1.00	1.01	1.00
	(0.70, 1.12)	(0.90, 1.06)	(0.92, 1.14)	(0.96, 1.12)	(0.97, 1.07)
BRidge	0.85	1.07	1.01	1.05	1.06
	(0.67, 1.08)	(0.93, 1.19)	(0.86, 1.18)	(0.91, 1.22)	(0.96, 1.19)
LASSO CV	0.83	1.00	0.99	1.00	1.00
	(0.69, 1.05)	(0.90, 1.06)	(0.90, 1.06)	(0.96, 1.07)	(0.97, 1.05)
BLASSO	0.84	1.00	0.98	1.00	1.00
	(0.69, 1.00)	(0.91, 1.03)	(0.87, 1.04)	(0.98, 1.02)	(0.99, 1.02)
Enet CV	0.77	0.99	0.95	0.98	0.98
	(0.55, 0.96)	(0.89, 1.03)	(0.85, 1.00)	(0.92, 1.01)	(0.92, 1.01)
SVR	0.92	1.03	0.99	1.02	1.03
	(0.85, 0.99)	(0.91, 1.11)	(0.92, 1.06)	(0.93, 1.11)	(0.95, 1.14)
RF	0.94	1.08	1.04	1.06	1.07
	(0.80, 1.13)	(0.94, 1.22)	(0.92, 1.14)	(0.95, 1.21)	(0.94, 1.19)
BART	0.91	1.06	1.00	1.01	1.02
	(0.77, 1.06)	(0.93, 1.22)	(0.89, 1.11)	(0.94, 1.11)	(0.95, 1.11)
Bagging	1.00	1.14	1.10	1.10	1.11
	(0.84, 1.22)	(0.98, 1.32)	(0.94, 1.24)	(0.96, 1.23)	(0.98, 1.27)
kNN	1.00	1.04	1.02	1.03	1.03
	(0.96, 1.06)	(0.92, 1.12)	(0.95, 1.11)	(0.96, 1.10)	(0.96, 1.13)
Huber	1.06	2.25	1.87	2.07	2.18

Table 5 – Confidence intervals for the MSE. For each model, the first line contains the average MSE and the second one, the limits of the interval. Some forecasting periods were omitted for the sake of simplicity. The results for these additional periods are similar to those reported in the table.

Model	**1**	**3**	**3M**	**6M**	**12M**
	(0.73, 1.42)	(1.67, 3.11)	(1.43, 2.40)	(1.58, 2.75)	(1.60, 2.89)
Theil-Sen	1.07	2.36	1.92	2.13	2.24
	(0.73, 1.45)	(1.76, 3.28)	(1.48, 2.45)	(1.60, 2.81)	(1.64, 2.98)
Factors	1.02	1.04	1.06	1.06	1.07
	(0.78, 1.31)	(0.89, 1.19)	(0.96, 1.21)	(0.95, 1.21)	(0.92, 1.25)
GARCH	0.71	1.19	0.75	0.71	0.73
	(0.27, 1.39)	(0.43, 2.40)	(0.51, 1.10)	(0.47, 1.05)	(0.46, 1.07)
VECM	0.71	1.71	0.92	0.98	1.10
	(0.47, 0.95)	(1.09, 2.69)	(0.59, 1.30)	(0.53, 1.48)	(0.38, 1.94)
SETAR	0.94	1.30	1.44	1.56	1.77
	(0.64, 1.28)	(0.82, 1.94)	(1.07, 1.93)	(1.11, 2.15)	(0.92, 2.44)
MA	0.93	1.12	1.12	1.06	1.02
	(0.66, 1.20)	(0.82, 1.52)	(0.76, 1.55)	(0.72, 1.51)	(0.70, 1.50)
SARIMA	0.77	1.15	0.75	0.70	0.70
	(0.58, 0.97)	(0.81, 1.55)	(0.53, 1.02)	(0.51, 0.98)	(0.43, 1.00)
ARFIMA	0.75	1.20	0.71	0.72	0.72
	(0.57, 1.02)	(0.87, 1.64)	(0.51, 0.96)	(0.49, 0.96)	(0.48, 1.03)
GradBoost	0.92	1.14	1.07	1.08	1.10
	(0.75, 1.12)	(0.93, 1.32)	(0.94, 1.22)	(0.95, 1.23)	(0.98, 1.27)
AdaBoost	0.93	1.06	1.04	1.06	1.07
	(0.82, 1.05)	(0.94, 1.18)	(0.92, 1.16)	(0.96, 1.18)	(0.96, 1.20)
Bayes Reg.	1.10	1.32	1.31	1.32	1.35
	(0.79, 1.51)	(1.05, 1.55)	(1.04, 1.70)	(1.07, 1.67)	(1.06, 1.67)

Table 6 – Average MSE reduction delivered by each model with respect to random walk. Negative values mean that the model increased the MSE. The winning model is highlighted in bold font.

Model	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
LSTM	83.6%	72.8%	72.4%	71.6%	70.9%	78.4%	71.4%	75.2%
ConvLSTM	**91.5%**	**79.0%**	**77.9%**	**77.9%**	**77.0%**	**83.4%**	**77.8%**	**82.8%**
MLP	50.5%	41.5%	42.5%	42.3%	40.8%	44.8%	43.7%	43.4%
RW	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Ridge CV	57.5%	49.3%	49.9%	49.9%	49.0%	50.6%	51.3%	51.7%
BRidge	57.7%	46.0%	46.9%	46.6%	45.7%	49.8%	49.1%	48.9%

Table 6 – Average MSE reduction delivered by each model with respect to random walk. Negative values mean that the model increased the MSE. The winning model is highlighted in bold font.

Model	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
LASSO CV	58.8%	49.3%	50.3%	50.4%	49.1%	51.2%	51.4%	51.8%
BLASSO	58.5%	49.5%	50.3%	50.5%	49.3%	51.6%	51.7%	51.9%
Enet CV	62.0%	50.1%	50.8%	50.9%	50.0%	53.0%	52.7%	52.7%
SVR	54.5%	48.1%	48.9%	49.1%	47.5%	50.7%	50.8%	50.4%
RF	53.3%	44.8%	46.1%	45.9%	45.0%	48.4%	48.6%	48.8%
BART	55.0%	47.1%	47.4%	47.2%	46.2%	50.4%	50.9%	51.1%
Bagging	50.4%	41.4%	43.2%	42.5%	41.5%	45.6%	46.9%	46.5%
kNN	50.3%	47.9%	48.6%	48.4%	47.4%	49.3%	50.2%	50.4%
Huber	47.4%	-12.5%	-11.7%	-11.9%	-13.4%	7.5%	-0.3%	-4.9%
Theil-Sen	47.0%	-18.4%	-17.5%	-17.7%	-19.0%	4.8%	-3.0%	-7.6%
Factors	49.6%	46.9%	48.2%	47.0%	46.2%	47.3%	48.5%	48.7%
GARCH	65.0%	46.0%	41.0%	40.1%	39.3%	63.0%	65.5%	64.8%
VECM	65.0%	25.7%	15.1%	17.2%	17.4%	54.4%	52.7%	47.3%
SETAR	53.6%	41.1%	35.3%	40.0%	38.6%	28.6%	24.2%	15.1%
MA	54.0%	43.1%	44.2%	46.8%	46.4%	44.4%	48.8%	51.2%
SARIMA	62.0%	47.8%	43.0%	40.7%	38.2%	63.0%	66.0%	66.6%
ARFIMA	62.7%	51.1%	40.5%	39.3%	38.6%	64.7%	65.3%	65.2%
GradBoost	54.6%	42.8%	43.3%	42.9%	42.2%	46.8%	47.5%	47.0%
AdaBoost	54.0%	46.6%	47.2%	47.2%	45.6%	48.4%	48.7%	48.6%
Bayes Reg.	45.5%	32.9%	34.3%	33.9%	31.4%	35.2%	35.9%	35.2%

Comparing the median MSE across the models, one infers that the winning model is the ConvLSTM coupled with a variational autoencoder for dimension reduction. Since this conclusion stems from a significant amount of simulations, with a rigorous division of the sample into training, validation, and test windows, the accomplishments of the ConvLSTM model are fairly sound and robust.

In fact, the superiority is statistically significant at 5% in most horizons considered and in comparison with most benchmarks, as confirmed by the confidence intervals computed. Usually, for more distant horizons, the intervals become wider due to the higher uncertainty and, thus, it is impossible to reject the null hypothesis that the models produce equivalent out-of-sample performance. Also, the standard LSTM model, despite exhibiting higher loss, is not far away from the ConvLSTM model, making it difficult to reject the null.

Still, given that the underlying properties of the ConvLSTM model confer enhanced flexibility to capture inflation dynamics and the strong results for short horizons, it is reasonable to claim that it should be superior in long horizons as well and that is yields more solid results relatively to the LSTM model. Furthermore, it must be mentioned that, in most machine learning studies, the direct comparison of the median (or average) MSE is sufficient to claim the superiority of a model and, using this criterion, the ConvLSTM is a indisputable winner.

By contrast, the random walk model is the worst performer in some forecasting windows, while Theil-Sen and Huber robust regressions, together with Bayesian linear regression, occupy the lowest ranks in other cases. An immediate conclusion is that, given the weak accuracy exhibited by the random walk, even simple econometric models are capable of enhancing out-of-sample predictions, and Table 6 confirms that the gains are expressive. Therefore, although inflation forecasting remains a challenging macroeconometric problem, the results shown here demonstrate that informative predictions can be generated, confronting other studies that claim the impossibility to beat random walk or naive moving averages. In particular, the ConvLSTM model generate loss reductions as large as 91.5% for some horizons.

In addition, corroborating the prior statement, the simple moving average model of Atkeson and Ohanian (2001) is also defeated by the ConvLSTM model and some of the benchmarks, proving that, in spite of the poor performance displayed by the standard linear Phillips curve in practice, inflation, although nonlinear, is predictable. However, accurate forecasts demand sophisticated nonlinear models and an extensive dataset of macroeconomic variables, explaining why some of the past studies on the subject failed to design reliable prediction models.

Analyzing the full picture, some interesting patterns emerge, and contrasting with the results reported by Medeiros et al. (2019) is fruitful due to the fact that they applied the same dataset as ours in their numerical exercise. Overall, corroborating their findings, it is also identified that machine learning methods and models that impose sparsity and/or regularization perform satisfactorily and provide substantial improvements with respect to the random walk.

However, also in accordance with Medeiros et al. (2019), deep learning models do not perform equally. Indeed, the multilayer perceptron delivers lackluster results, consistently with the findings by Medeiros et al. (2019), who train a deep neural network with three hidden layers and 32, 16, and 8 ReLU neurons in each layer, respectively, and observe that such model is outperformed by Random Forests, for instance. In our view, such behavior is explained by the fact that, unlike LSTM and ConvLSTM, the MLP does not have an architecture designed to capture temporal dependencies in the input data. Therefore, it requires an excessive number of parameters to adjust to the inputs, leading to overfitting and, thus, poor out-of-sample performance.

The performance of the factor model offers interesting insights. The mixed results are most likely explained by a missing structure of linear factors in the data, consistently with the findings reported by Medeiros et al. (2019). This conclusion is also supported by the fragile accuracy of other standard linear models added as benchmarks and justifies the use of deep learning to identify these nonlinear interactions between factors.

Remarkably, in a related study using financial data and autoencoders, Gu, Kelly and Xiu (2020) find that nonlinearities are crucial for extracting factors and improving their explanatory power. Given the nature and similarities between their dataset and the one considered in this work, it is reasonable to believe that nonlinearities are relevant to explain the findings herein. The reported benefits of adding a variational autoencoder also reinforce this view.

Turning the attention to the nonlinearities in inflation per se, the superiority of models such as ConvLSTM, LSTM and Random Forests with respect to the random walk is consistent with the claim that inflation is governed by a nonlinear stochastic process, ruling out the possibility of modeling inflation with the standard, linear Phillips curve. However, LASSO, Ridge and Elastic Net, which are linear models, also delivered decent performance, although inferior to ConvLSTM and LSTM.

This means that, despite the evidence of (unknown) nonlinearity, even flexible, nonparametric models such as the Random Forest may fail to beat more simple ones. Hence, one may infer that the nonlinear relationship between inflation and other macroeconomic variables is far from trivial, and such complexity can only be effectively addressed by deep learning models, which are capable of learning them with supervision. Moreover, the outcomes of these models indicate that variable selection play an important role when large datasets are employed, which is the case here. Ultimately, it also another evidence that conventional linear models should be excluded from practical applications.

A question that immediately arises when adjusting models to time series is whether the superior performance of a certain model is verified exclusively in a particular time window, or whether it depends on business cycles or other exogenous variables. The approach established in this work, based on successive splits of the dataset, conceiving a diverse collection of training, validation, and test samples with which models are adjusted, addresses this question. Hence, each model has been trained on different samples, covering distinct periods of time, and tested on multiple settings as well.

Indeed, inspecting the confidence intervals estimated, all the performance metrics, including the MSE, unveil a considerable dispersion of results across simulations, meaning that different periods produce distinct ranks. This conclusion is consistent with those reported in other papers, such as Kim (1993), signaling the existence of regimes in inflation data. However, via these intervals, one can statistically claim the superiority of a given model, and, as the outputs corroborate, the hypothesis that the ConvLSTM network proposed, coupled with variational autoencoders, outperforms its benchmarks cannot be rejected.

With the purpose of inspecting how MSE behaves across time, Figure 32 displays the evolution of the MSE for the ConvLSTM. For the benchmarks, the behavior is similar, justifying the omission of the respective plots. Comparing with Figure 33, it is immediately noticed a strong, positive correlation between the MSE and the CPI volatility, meaning that, in periods of greater turbulence, performance seems to deteriorate, which is reasonable. Still, in the case of the model proposed, even during the 2008-09 crisis, when volatility reached its peak, the MSE remained under control. For the sake of brevity, the full results are not reported here, but the ones provided also indicate that the superiority of the ConvLSTM does not depend on the state of the economy, meaning that it outperforms both during expansions and recessions, or during periods with high and low uncertainty.

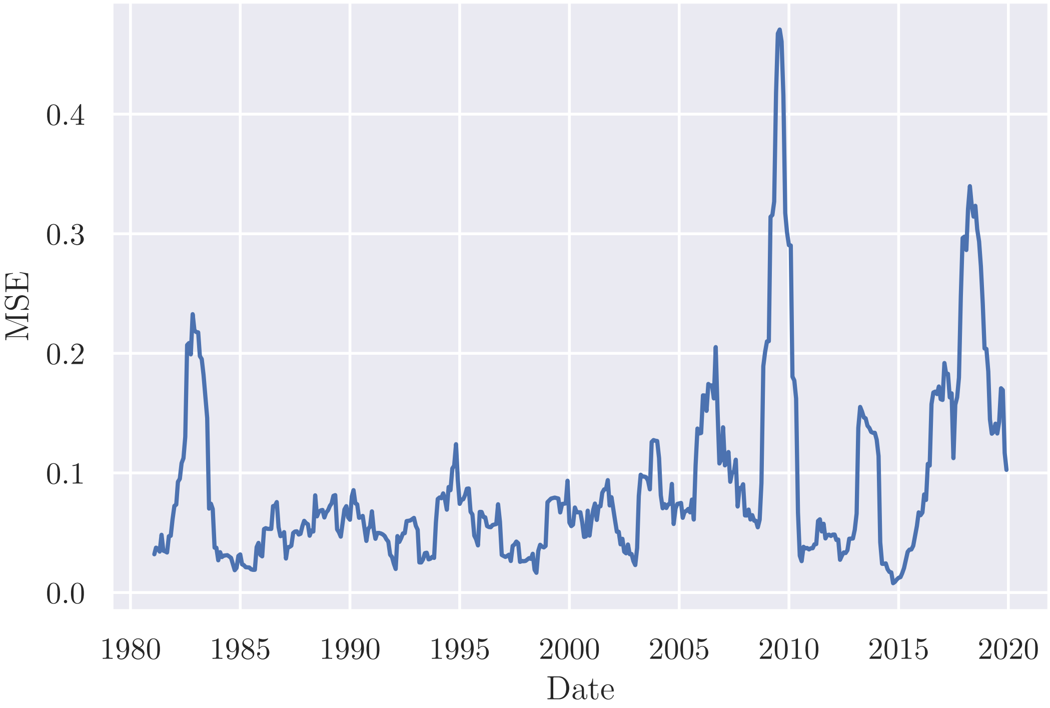


Figure 32 – MSE of the ConvLSTM model computed using a 12-month rolling window.

In the framework of machine learning and big data, dimension reduction gains become explicit when assessing the performance offered by the ConvLSTM model with and without the variational autoencoder. As the outcomes show, reducing the dimension of the dataset improves the out-of-sample performance, meaning that redundant information exist in the time series. In order to confirm that using a nonlinear technique for dimension reduction provides greater gains than a linear one, the same exercise is conducted, but replacing the VAE by PCA. The results demonstrate that the combination of VAE and ConvLSTM is superior to the alternative specification proposed, which is another piece of evidence of the existence of a nonlinear factor structure in the input data.

In addition to the outcomes presented and debated so far, Table 7, Table 8, Table 9,

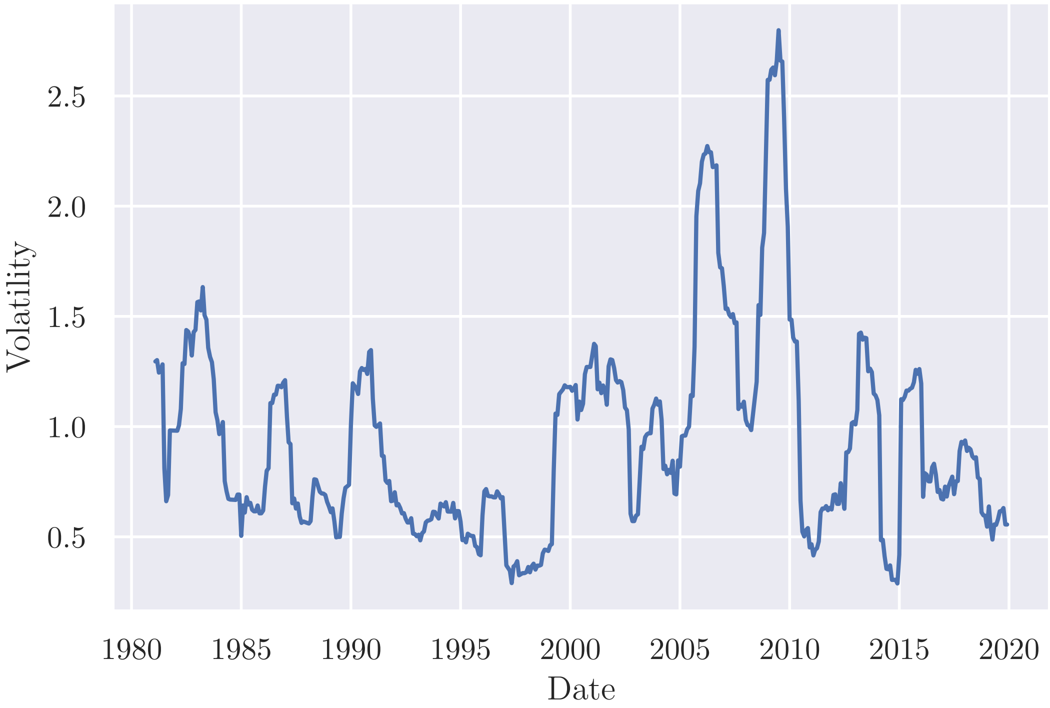


Figure 33 – Volatility of the normalized second differences of log prices, as measured by the CPI, using a 12-month rolling window.

and Table 10 display the MAE computed for each model and the corresponding statistics. Overall, the conclusions obtained are essentially the same offered by the MSE criterion, despite the fact that confidence intervals seem wider and losses are higher. Supplementary metrics, such as RMSE, MAPE and CS, have also been implemented in the routines mentioned in section 4.4 and can be easily replicated. They are also available upon request. In summary, these metrics produce results consistent with those discussed so far.

Table 7 – Descriptive statistics of the out-of-sample MAE of each model.

Model	**Statistic**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
LSTM	Min.	0.29	0.42	0.47	0.49	0.49	0.40	0.43	0.45
	Q1	0.40	0.55	0.57	0.57	0.56	0.49	0.55	0.51
	Median	0.45	0.58	0.60	0.59	0.61	0.53	0.59	0.57
	Q3	0.51	0.63	0.63	0.64	0.64	0.58	0.66	0.63
	Max.	0.72	0.84	0.79	1.03	0.91	0.70	0.88	0.86
ConvLSTM	Min.	0.25	0.45	0.48	0.50	0.50	0.37	0.40	0.33
	Q1	0.30	0.52	0.53	0.54	0.54	0.44	0.50	0.43
	Median	0.32	0.53	0.54	0.54	0.54	0.46	0.53	0.46
	Q3	0.35	0.54	0.55	0.55	0.55	0.49	0.58	0.50
	Max.	0.41	0.56	0.56	0.55	0.56	0.58	0.75	0.60

Table 7 – Descriptive statistics of the out-of-sample MAE of each model.

Model	**Statistic**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
MLP	Min.	0.63	0.71	0.69	0.68	0.67	0.72	0.74	0.78
	Q1	0.68	0.76	0.76	0.76	0.77	0.79	0.82	0.86
	Median	0.71	0.79	0.79	0.80	0.80	0.82	0.85	0.88
	Q3	0.74	0.82	0.82	0.84	0.83	0.85	0.88	0.91
	Max.	0.84	0.88	0.87	0.95	0.91	0.91	0.96	0.98
RW	Min.	0.91	0.88	0.92	0.93	0.91	0.91	0.93	0.96
	Q1	1.02	1.01	1.01	1.02	1.00	1.06	1.08	1.09
	Median	1.07	1.05	1.06	1.07	1.05	1.10	1.11	1.13
	Q3	1.10	1.09	1.11	1.10	1.11	1.13	1.15	1.16
	Max.	1.20	1.18	1.20	1.22	1.28	1.24	1.30	1.25
Ridge CV	Min.	0.56	0.64	0.64	0.63	0.63	0.67	0.71	0.74
	Q1	0.62	0.69	0.69	0.68	0.69	0.75	0.77	0.80
	Median	0.64	0.72	0.71	0.71	0.71	0.77	0.79	0.82
	Q3	0.67	0.73	0.73	0.74	0.74	0.79	0.82	0.84
	Max.	0.82	0.78	0.78	0.78	0.79	0.84	0.87	0.91
BRidge	Min.	0.55	0.67	0.65	0.66	0.66	0.69	0.73	0.74
	Q1	0.63	0.72	0.72	0.72	0.72	0.75	0.79	0.81
	Median	0.65	0.75	0.74	0.76	0.75	0.78	0.82	0.84
	Q3	0.68	0.78	0.77	0.78	0.78	0.80	0.84	0.86
	Max.	0.80	0.86	0.83	0.84	0.85	0.86	0.93	0.94
LASSO CV	Min.	0.58	0.64	0.65	0.63	0.64	0.68	0.70	0.72
	Q1	0.62	0.69	0.69	0.69	0.70	0.74	0.77	0.80
	Median	0.64	0.71	0.71	0.71	0.71	0.76	0.79	0.82
	Q3	0.66	0.73	0.73	0.74	0.74	0.78	0.82	0.84
	Max.	0.76	0.78	0.77	0.79	0.81	0.84	0.88	0.89
BLASSO	Min.	0.53	0.64	0.65	0.63	0.64	0.67	0.70	0.72
	Q1	0.63	0.69	0.69	0.68	0.69	0.74	0.77	0.80
	Median	0.66	0.71	0.71	0.71	0.71	0.76	0.79	0.82
	Q3	0.69	0.73	0.73	0.73	0.74	0.78	0.81	0.84
	Max.	0.75	0.77	0.77	0.78	0.79	0.83	0.86	0.89
Enet CV	Min.	0.55	0.64	0.64	0.64	0.64	0.66	0.70	0.72
	Q1	0.60	0.68	0.68	0.68	0.68	0.73	0.76	0.79
	Median	0.63	0.71	0.71	0.71	0.71	0.75	0.78	0.81
	Q3	0.65	0.73	0.72	0.73	0.73	0.77	0.81	0.83
	Max.	0.74	0.77	0.77	0.78	0.79	0.81	0.86	0.87
SVR	Min.	0.59	0.64	0.66	0.65	0.65	0.68	0.73	0.74
	Q1	0.65	0.70	0.70	0.70	0.70	0.74	0.78	0.80

Table 7 – Descriptive statistics of the out-of-sample MAE of each model.

Model	**Statistic**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
	Median	0.67	0.73	0.73	0.73	0.73	0.77	0.80	0.83
	Q3	0.69	0.75	0.74	0.75	0.75	0.79	0.82	0.85
	Max.	0.76	0.81	0.79	0.81	0.83	0.82	0.88	0.90
RF	Min.	0.55	0.66	0.67	0.65	0.67	0.69	0.73	0.76
	Q1	0.67	0.74	0.72	0.73	0.73	0.76	0.79	0.81
	Median	0.69	0.76	0.75	0.76	0.76	0.79	0.82	0.84
	Q3	0.71	0.78	0.77	0.78	0.79	0.81	0.84	0.87
	Max.	0.79	0.83	0.87	0.87	0.92	0.86	0.92	0.92
BART	Min.	0.60	0.64	0.66	0.64	0.66	0.67	0.71	0.74
	Q1	0.66	0.71	0.71	0.71	0.72	0.75	0.78	0.80
	Median	0.68	0.74	0.74	0.74	0.75	0.77	0.80	0.83
	Q3	0.71	0.76	0.76	0.77	0.78	0.79	0.82	0.85
	Max.	0.80	0.81	0.83	0.83	0.88	0.85	0.88	0.91
Bagging	Min.	0.65	0.70	0.67	0.69	0.69	0.72	0.71	0.73
	Q1	0.70	0.77	0.74	0.75	0.76	0.79	0.81	0.83
	Median	0.72	0.79	0.77	0.78	0.79	0.82	0.83	0.86
	Q3	0.74	0.82	0.80	0.81	0.82	0.83	0.86	0.89
	Max.	0.83	0.90	0.89	0.88	0.91	0.90	0.92	0.95
kNN	Min.	0.63	0.66	0.65	0.62	0.66	0.67	0.72	0.73
	Q1	0.69	0.71	0.70	0.71	0.71	0.76	0.78	0.81
	Median	0.71	0.73	0.72	0.73	0.73	0.78	0.81	0.83
	Q3	0.73	0.75	0.75	0.76	0.77	0.80	0.83	0.85
	Max.	0.78	0.79	0.80	0.82	0.83	0.84	0.90	0.91
Huber	Min.	0.63	0.99	0.94	0.94	0.96	0.92	0.92	0.99
	Q1	0.72	1.10	1.09	1.11	1.10	1.01	1.09	1.11
	Median	0.76	1.15	1.15	1.17	1.16	1.07	1.14	1.16
	Q3	0.79	1.22	1.22	1.23	1.25	1.10	1.20	1.24
	Max.	0.87	1.46	1.38	1.45	1.43	1.24	1.37	1.41
Theil-Sen	Min.	0.63	1.01	0.95	0.99	1.00	0.94	0.92	1.00
	Q1	0.72	1.11	1.12	1.13	1.13	1.02	1.09	1.12
	Median	0.76	1.19	1.18	1.20	1.20	1.08	1.16	1.18
	Q3	0.79	1.25	1.26	1.27	1.28	1.12	1.22	1.26
	Max.	0.87	1.48	1.41	1.50	1.48	1.25	1.38	1.44
Factors	Min.	0.61	0.65	0.63	0.66	0.65	0.71	0.74	0.74
	Q1	0.69	0.71	0.71	0.71	0.71	0.76	0.79	0.81
	Median	0.72	0.74	0.74	0.74	0.74	0.79	0.82	0.83
	Q3	0.76	0.76	0.76	0.77	0.77	0.82	0.85	0.86

Table 7 – Descriptive statistics of the out-of-sample MAE of each model.

Model	Statistic	1	2	3	6	12	3M	6M	12M
GARCH	Max.	0.85	0.83	0.81	1.50	0.88	0.90	0.95	1.32
	Min.	0.38	0.49	0.48	0.49	0.49	0.53	0.50	0.50
	Q1	0.56	0.67	0.68	0.70	0.71	0.62	0.61	0.62
	Median	0.63	0.74	0.79	0.81	0.81	0.67	0.65	0.67
	Q3	0.72	0.89	0.93	0.93	0.95	0.71	0.70	0.73
VECM	Max.	0.99	1.20	1.24	1.33	1.39	0.86	0.86	0.94
	Min.	0.48	0.71	0.80	0.69	0.81	0.59	0.54	0.49
	Q1	0.61	0.89	0.95	0.90	0.94	0.70	0.69	0.72
	Median	0.64	0.93	1.02	0.96	0.99	0.75	0.78	0.85
	Q3	0.68	1.02	1.12	1.03	1.06	0.81	0.86	0.94
SETAR	Max.	0.77	1.16	1.36	1.24	1.46	0.97	1.04	1.20
	Min.	0.51	0.62	0.64	0.65	0.60	0.78	0.79	0.65
	Q1	0.64	0.73	0.78	0.76	0.75	0.87	0.92	1.00
	Median	0.68	0.78	0.85	0.81	0.81	0.93	0.98	1.07
	Q3	0.72	0.84	0.91	0.87	0.89	0.99	1.06	1.14
MA	Max	0.82	0.95	1.06	0.99	0.98	1.13	1.21	1.39
	Min.	0.48	0.52	0.51	0.49	0.49	0.53	0.50	0.48
	Q1	0.61	0.68	0.66	0.64	0.64	0.68	0.64	0.62
	Median	0.68	0.76	0.75	0.73	0.74	0.75	0.73	0.72
	Q3	0.76	0.85	0.85	0.84	0.86	0.83	0.83	0.84
SARIMA	Max.	0.97	1.10	1.10	1.09	1.12	1.07	1.07	1.09
	Min.	0.56	0.64	0.68	0.74	0.72	0.56	0.57	0.50
	Q1	0.67	0.77	0.81	0.83	0.84	0.65	0.62	0.62
	Median	0.71	0.81	0.85	0.88	0.88	0.69	0.67	0.67
	Q3	0.74	0.85	0.91	0.93	0.95	0.73	0.71	0.71
ARFIMA	Max.	0.84	1.00	1.04	1.10	1.09	0.84	0.81	0.93
	Min.	0.58	0.61	0.72	0.70	0.73	0.53	0.51	0.52
	Q1	0.65	0.73	0.80	0.82	0.83	0.63	0.64	0.63
	Median	0.69	0.77	0.85	0.87	0.87	0.66	0.68	0.68
	Q3	0.72	0.84	0.94	0.92	0.93	0.70	0.71	0.72
GradBoost	Max.	0.86	0.93	1.07	1.12	1.07	0.83	0.80	0.83
	Min.	0.59	0.68	0.70	0.71	0.67	0.71	0.75	0.74
	Q1	0.66	0.75	0.76	0.75	0.75	0.78	0.80	0.83
	Median	0.68	0.77	0.78	0.78	0.78	0.80	0.83	0.86
	Q3	0.71	0.80	0.80	0.80	0.81	0.82	0.85	0.88
AdaBoost	Max.	0.77	0.89	0.86	0.88	0.92	0.90	0.91	0.93
	Min.	0.60	0.66	0.66	0.67	0.65	0.69	0.73	0.75

Table 7 – Descriptive statistics of the out-of-sample MAE of each model.

Model	**Statistic**	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
	Q1	0.66	0.72	0.72	0.72	0.72	0.76	0.79	0.82
	Median	0.69	0.74	0.74	0.74	0.75	0.78	0.82	0.84
	Q3	0.71	0.76	0.77	0.77	0.78	0.81	0.84	0.86
	Max.	0.80	0.83	0.81	0.83	0.87	0.89	0.93	0.93
Bayes Reg.	Min.	0.65	0.71	0.74	0.72	0.73	0.75	0.76	0.77
	Q1	0.73	0.82	0.82	0.82	0.83	0.85	0.86	0.90
	Median	0.77	0.86	0.86	0.87	0.87	0.88	0.91	0.93
	Q3	0.81	0.90	0.90	0.90	0.92	0.92	0.95	0.98
	Max.	0.93	1.01	1.01	1.07	1.05	1.06	1.08	1.09

Table 8 – Ranks produced by comparing the models according to their average MAE through the simulations. The two best models are highlight in bold font.

Model	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
LSTM	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**
ConvLSTM	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
MLP	20	20	17	17	19	20	21	21
RW	26	24	24	24	24	26	24	24
Ridge CV	6	6	6	6	6	11	11	10
BRidge	8	13	11	13	13	15	17	17
LASSO CV	7	3	5	5	5	10	9	9
BLASSO	9	5	3	3	4	9	10	8
Enet CV	3	4	4	4	3	7	8	7
SVR	10	7	8	7	8	13	13	12
RF	15	15	14	14	15	17	18	15
BART	13	11	10	11	10	12	12	11
Bagging	21	19	15	16	17	21	19	20
kNN	19	8	7	9	7	14	14	13
Huber	23	25	25	25	25	24	25	25
Theil-Sen	24	26	26	26	26	25	26	26
Factors	22	9	9	10	11	18	15	14
GARCH	4	12	18	19	12	4	3	4
VECM	5	23	23	23	23	8	7	18
SETAR	11	18	19	18	18	23	23	23
MA	14	14	13	8	9	6	6	6
SARIMA	18	21	21	22	21	5	4	3
ARFIMA	16	16	20	21	20	3	5	5
GradBoost	12	17	16	15	16	19	20	19

Table 8 – Ranks produced by comparing the models according to their average MAE through the simulations. The two best models are highlight in bold font.

Model	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
AdaBoost	17	10	12	12	14	16	16	16
Bayes Reg.	25	22	22	20	22	22	22	22

Table 9 – Confidence intervals for the MAE. For each model, the first line contains the average MAE and the second one, the limits of the interval. Some forecasting periods were omitted for the sake of simplicity. The results for these additional periods are similar to those reported in the table.

Model	**1**	**3**	**3M**	**6M**	**12M**
LSTM	0.46	0.60	0.53	0.61	0.58
	(0.29, 0.67)	(0.51, 0.73)	(0.41, 0.67)	(0.48, 0.85)	(0.46, 0.77)
ConvLSTM	0.33	0.54	0.46	0.54	0.47
	(0.28, 0.40)	(0.51, 0.55)	(0.38, 0.55)	(0.42, 0.68)	(0.37, 0.58)
MLP	0.71	0.79	0.82	0.86	0.88
	(0.63, 0.80)	(0.71, 0.86)	(0.73, 0.90)	(0.77, 0.95)	(0.79, 0.97)
RW	1.06	1.06	1.10	1.12	1.12
	(0.93, 1.18)	(0.94, 1.18)	(0.95, 1.20)	(1.00, 1.24)	(1.00, 1.25)
Ridge CV	0.65	0.71	0.77	0.79	0.82
	(0.59, 0.79)	(0.65, 0.77)	(0.70, 0.83)	(0.73, 0.86)	(0.75, 0.88)
BRidge	0.66	0.74	0.78	0.81	0.84
	(0.58, 0.77)	(0.67, 0.81)	(0.69, 0.85)	(0.74, 0.90)	(0.75, 0.92)
LASSO CV	0.64	0.71	0.76	0.79	0.82
	(0.58, 0.72)	(0.65, 0.77)	(0.70, 0.83)	(0.73, 0.86)	(0.75, 0.88)
BLASSO	0.66	0.71	0.76	0.79	0.82
	(0.58, 0.75)	(0.65, 0.77)	(0.69, 0.82)	(0.73, 0.86)	(0.75, 0.88)
Enet CV	0.63	0.70	0.75	0.79	0.81
	(0.56, 0.70)	(0.65, 0.76)	(0.68, 0.80)	(0.73, 0.85)	(0.74, 0.87)
SVR	0.67	0.72	0.77	0.80	0.83
	(0.62, 0.74)	(0.67, 0.78)	(0.70, 0.82)	(0.73, 0.88)	(0.75, 0.90)
RF	0.69	0.75	0.79	0.82	0.84
	(0.63, 0.79)	(0.69, 0.83)	(0.71, 0.85)	(0.75, 0.90)	(0.77, 0.91)
BART	0.68	0.74	0.77	0.80	0.82
	(0.61, 0.76)	(0.67, 0.81)	(0.68, 0.84)	(0.73, 0.87)	(0.74, 0.90)
Bagging	0.72	0.78	0.81	0.83	0.86
	(0.65, 0.80)	(0.70, 0.87)	(0.74, 0.89)	(0.74, 0.90)	(0.77, 0.93)
kNN	0.71	0.73	0.78	0.81	0.83
	(0.64, 0.77)	(0.65, 0.79)	(0.70, 0.84)	(0.73, 0.87)	(0.75, 0.90)

Table 9 – Confidence intervals for the MAE. For each model, the first line contains the average MAE and the second one, the limits of the interval. Some forecasting periods were omitted for the sake of simplicity. The results for these additional periods are similar to those reported in the table.

Model	**1**	**3**	**3M**	**6M**	**12M**
Huber	0.75	1.16	1.07	1.14	1.17
	(0.65, 0.85)	(1.01, 1.35)	(0.94, 1.22)	(0.97, 1.33)	(1.00, 1.38)
Theil-Sen	0.76	1.19	1.08	1.16	1.19
	(0.65, 0.86)	(1.03, 1.39)	(0.95, 1.23)	(0.99, 1.34)	(1.01, 1.39)
Factors	0.73	0.73	0.79	0.82	0.84
	(0.63, 0.83)	(0.65, 0.80)	(0.73, 0.88)	(0.75, 0.90)	(0.75, 0.93)
GARCH	0.64	0.81	0.67	0.66	0.67
	(0.41, 0.88)	(0.52, 1.11)	(0.55, 0.84)	(0.52, 0.81)	(0.51, 0.81)
VECM	0.64	1.03	0.76	0.78	0.83
	(0.54, 0.76)	(0.83, 1.31)	(0.61, 0.92)	(0.57, 1.00)	(0.50, 1.13)
SETAR	0.67	0.84	0.93	0.99	1.06
	(0.55, 0.78)	(0.66, 1.03)	(0.79, 1.11)	(0.82, 1.18)	(0.77, 1.29)
MA	0.68	0.75	0.75	0.73	0.72
	(0.59, 0.78)	(0.62, 0.90)	(0.63, 0.88)	(0.58, 0.89)	(0.55, 0.90)
SARIMA	0.70	0.86	0.69	0.67	0.67
	(0.60, 0.80)	(0.71, 1.01)	(0.58, 0.83)	(0.57, 0.81)	(0.52, 0.82)
ARFIMA	0.69	0.87	0.67	0.68	0.68
	(0.59, 0.80)	(0.73, 1.06)	(0.57, 0.79)	(0.56, 0.79)	(0.54, 0.83)
GradBoost	0.68	0.78	0.80	0.83	0.85
	(0.61, 0.75)	(0.70, 0.85)	(0.73, 0.89)	(0.76, 0.90)	(0.77, 0.93)
AdaBoost	0.69	0.74	0.79	0.82	0.84
	(0.61, 0.76)	(0.68, 0.80)	(0.72, 0.86)	(0.75, 0.89)	(0.77, 0.92)
Bayes Reg.	0.77	0.86	0.89	0.91	0.93
	(0.66, 0.92)	(0.77, 0.96)	(0.77, 1.03)	(0.79, 1.04)	(0.80, 1.05)

Table 10 – Average MAE reduction delivered by each model with respect to random walk. Negative values mean that the model increased the MSE. The winning model is highlighted in bold font.

Model	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
LSTM	57.9%	44.5%	43.5%	43.5%	42.1%	51.6%	46.9%	49.5%
ConvLSTM	**69.7%**	**49.7%**	**48.9%**	**49.2%**	**48.0%**	**58.2%**	**52.1%**	**58.8%**
MLP	33.1%	24.5%	25.7%	25.6%	23.8%	25.7%	23.3%	22.0%
RW	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Ridge CV	39.8%	31.6%	32.9%	33.1%	31.8%	30.2%	28.8%	27.1%

Table 10 – Average MAE reduction delivered by each model with respect to random walk. Negative values mean that the model increased the MSE. The winning model is highlighted in bold font.

Model	**1**	**2**	**3**	**6**	**12**	**3M**	**6M**	**12M**
BRidge	39.1%	28.7%	30.1%	29.9%	28.4%	29.2%	26.5%	25.3%
LASSO CV	39.7%	32.6%	33.7%	33.4%	32.3%	30.4%	28.9%	27.3%
BLASSO	38.3%	32.4%	33.7%	33.8%	32.4%	30.6%	28.9%	27.3%
Enet CV	41.2%	32.6%	33.7%	33.7%	32.3%	31.4%	29.6%	27.7%
SVR	36.9%	30.4%	31.6%	32.5%	30.4%	29.7%	28.1%	26.1%
RF	35.7%	27.4%	29.3%	29.7%	27.1%	28.5%	26.4%	25.6%
BART	36.3%	29.2%	30.5%	30.5%	28.8%	30.0%	28.5%	26.6%
Bagging	32.5%	24.6%	27.4%	26.8%	24.7%	25.6%	25.7%	23.6%
kNN	33.2%	30.3%	32.0%	31.6%	30.4%	29.4%	27.6%	26.0%
Huber	29.2%	-10.0%	-7.8%	-9.0%	-11.2%	2.5%	-2.0%	-2.9%
Theil-Sen	28.6%	-14.0%	-10.6%	-11.9%	-14.5%	1.4%	-4.2%	-4.6%
Factors	32.2%	29.8%	30.9%	30.5%	29.2%	27.6%	26.7%	26.0%
GARCH	40.7%	29.1%	25.4%	23.8%	23.0%	39.3%	41.4%	40.6%
VECM	39.9%	11.2%	3.8%	5.9%	5.4%	31.3%	29.8%	24.6%
SETAR	36.3%	25.5%	20.4%	24.6%	23.0%	15.7%	12.3%	5.3%
MA	35.9%	27.9%	29.9%	31.9%	29.6%	32.0%	34.3%	36.5%
SARIMA	33.7%	22.4%	19.9%	18.1%	15.9%	37.2%	40.3%	40.9%
ARFIMA	35.4%	27.0%	20.1%	18.6%	17.1%	39.5%	38.8%	39.8%
GradBoost	36.3%	26.1%	26.7%	27.2%	26.0%	27.4%	25.6%	24.0%
AdaBoost	35.3%	29.4%	30.1%	30.0%	28.8%	28.7%	26.6%	25.6%
Bayes Reg.	27.7%	18.0%	18.9%	19.6%	17.3%	19.7%	18.4%	17.3%

In conclusion, the results shown in this section confirm that the combination of ConvLSTM and variational autoencoders yields the best out-of-sample performance according to different metrics and in distinct windows of the time horizon considered in this study.

6 CONCLUSIONS

The present work is devoted to the evaluation of deep learning methods for inflation forecasting, a daunting and unsolved problem that has challenged academics and practitioners in the past decades. In particular, a combination of ConvLSTM networks and variational autoencoders is proposed due to the success these models have demonstrated in several practical applications involving time series analysis, as the pertaining literature unveils. With the purpose of corroborating the superiority of the advocated combination of models, a wide selection of benchmarks comprised by popular econometric and machine learning models is adjusted to US inflation data. In the sequence, an out-of-sample performance comparison is implemented to determine the winning model.

As expected, the experiments demonstrate that coupling variational autoencoders to a ConvLSTM networks yields compelling results. Merging these techniques significantly improves the out-of-sample accuracy in comparison with the benchmarks, generating the lowest median MSE across simulations. The same conclusion is obtained using distinct performance metrics, such as MAE, and is robust in multiple forecasting horizons. The simulations using different training and test samples confirm that, despite the variations observed across iterations, the proposed model delivers more accurate predictions in every scenario.

Furthermore, by demonstrating that deep learning is more effective when tackling the challenges of inflation forecasting that emerge due to nonlinearities and nonstationarity, as confirmed by empirical evidences, the present work offers additional proof that linear models such as the standard Phillips curve are inadequate to explain inflation dynamics. It is also attested that inflation is explained by multiple macroeconomic variables, most of which are highly correlated, and employing techniques for dimension reduction such as the variational autoencoder improves the quality of the forecasts.

The findings reported here also suggest that similar accomplishments may be achieved using deep learning for modeling and forecasting other macroeconomic variables. Indeed, some reasons presented here to justify the existence of a nonlinear dynamics governing inflation are valid for other macroeconomic time series. Thereby, further studies may be conducted to investigate the application of deep networks in these contexts, comparing the results with methods traditionally adopted to model these series.

Additionally, nonlinearities seem to abound in the macroeconomic dataset contemplated in this work. This conclusion stems from the fact that the variational autoencoder managed to improve the performance of the ConvLSTM model, while linear factor models displayed mediocre out-of-sample accuracy. Together, these findings suggest that a nonlinear factor structure governs these macroeconomic variables, in agreement with the study by Gu, Kelly and Xiu (2020), who detect nonlinear factors in a large set of financial time series through the application of autoencoders. Given the analogous nature of these sets and the deep connection between financial and macroeconomic variables, it is reasonable to believe that a similar structure should be present in the dataset considered herein. A thorough examination of this aspect would be pertinent for future projects, giving an important contribution to the literature.

Naturally, given that deep learning is a ever-growing field of research, with many discoveries happening continuously, multiple extensions of the present work can be envisioned. For instance, in what concerns the transformation of the input data for denoising and dimension reduction, wavelets have attracted attention in recent years for time series denoising and decomposition into uncorrelated components (PERCIVAL; WALDEN, 2006).

In practice, the discrete version of the wavelet transform (DWT) is more common for time series, since these operate in discrete time as well. As highlighted by Percival and Walden (2006), DWT rewrites a time series in terms of coefficients associated with a particular time employing a dyadic scale. This decomposition by time scale allow wavelets to deal with nonstationary series, which are frequently observed in Economics and Finance (HSIEH; HSIAO; YEH, 2011). By working with multiple scales and high resolution, wavelets determine not only which frequencies are existent in a signal, but also at which time they have occurred, which is helpful to detect noise and local features.

An instructive example of wavelets applied when forecasting stock markets is developed by Hsieh, Hsiao and Yeh (2011). The authors introduce an integrated system where wavelet transforms and recurrent neural networks are merged for stock price forecasting. In this process, wavelets are implemented to decompose stock price time series and, thus, eliminate noise, yielding promising results. In a similar fashion, Gallegati (2008) employs discrete wavelet transforms to the Dow Jones stock index and the industrial production index for the US to scrutinize the ability of the stock market to anticipate the future level of economic activity. Likewise, Chang, Zhang and Chen (2019) combine LSTM networks and wavelet transforms in the context of electricity price prediction. The authors report compelling improvements in forecasting accuracy over simpler models. Ultimately, Bao, Yue and Rao (2017) exploit DWT for decomposing and denoising stock price series and later model the transformed data using a combination of stacked autoencoders and LSTM networks.

Besides, alternative architectures for the ConvLSTM and VAE adopted in this work could also be scrutinized to optimize out-of-sample performance. For instance, in terms of activation function, an interesting option is the *Swish* function discovered by Ramachandran, Zoph and Le (2018). It is defined by:

$$f(x) = x \cdot \sigma(\beta x) \tag{6.1}$$

where $\sigma(\cdot)$ is the sigmoid function and β is a parameter. Experiments show that Swish outperforms ReLU on deeper networks across a number of challenging datasets. Also, distinct architectures for the LSTM part of the model that could be tested are provided by

Yu et al. (2019), while, with respect to autoencoders, Dong et al. (2018) provide auspicious alternatives.

Another area to which significant time was dedicated and extensions are feasible comprehends the prevention of overfitting. In this aspect, a notorious criticism with respect to neural networks is that these models often lack transparency, are highly parameterized, and have complex interpretations due to the existence of many nonlinear interactions between their parts. Hence, they are judged as black boxes, and their out-of-sample accuracy is often questioned. Thus, by ensuring that overfitting is avoided, more confidence is obtained regarding generalization and out-of-sample performance. Here, techniques such as dropout layers and batch normalization were extensively used in order to address this problem. Still, alternatives exist. For instance, Courbariaux, Bengio and David (2015) introduce the BinaryConnect, a method which consists in training a deep neural network with binary weights. The authors conclude that, analogously to other dropout schemes, BinaryConnect acts as a regularizer and yields near state-of-the-art results in many applications.

Moreover, out-of-sample performance analysis could also be improved using generative adversarial networks (GAN), introduced in the seminal paper by Goodfellow et al. (2014). Essentially, in these models, two networks are simultaneously trained: a generative one, G, that captures the data distribution, and a discriminative one, D, that estimates the probability that a sample came from the training data rather than from the generative network. G is trained to maximize its ability to fool D. Such framework is equivalent to a minimax two-player game popular in Game Theory.

An application involving GANs in the context of inflation forecasting would be the generation of additional datasets, enriching the out-of-sample evaluation of the models tested here. As an illustration, Takahashi, Chen and Tanaka-Ishii (2019) use GANs for financial time series, obtaining promising results. The authors demonstrate that the GAN model learns the properties of data and produces realistic time series in a data-driven manner, replicating statistical properties such as linear unpredictability, heavy tails, volatility clustering, leverage effects, and asymmetry.

The detailed and profound analysis of which macroeconomic variables should be added or excluded when forecasting inflation was outside the scope of this work, which focused on demonstrating the benefits of using deep learning in this context. Indeed, this delimitation is one of the factors justifying the decision to employ the dataset provided by McCracken and Ng (2016), who rigorously follow the best practices in the literature when compiling the data to ensure that it could be used in academic research. Nevertheless, this decision does not imply that variable selection could not improve out-of-sample performance.

An encouraging approach comprehends the use of disaggregated prices that could be individually forecast for later combination to predict the aggregated CPI. The literature shows that interesting results can be found with these time series. For instance, Monacelli and Sala (2009) estimate the contribution of international common factors to the dynamics of price inflation rates of a cross-section of 948 CPI products in US, Germany, France, and UK. The authors conclude that one international common factor explains between 15% and 30% of the variance of consumer prices between 1991 and 2004. Besides, they identify a positive and statistically significant relationship between exposure of consumer inflation to international shocks and trade openness at the sectoral level. Coleman (2010) also utilizes disaggregated prices in sub-Saharan Africa, where inflation persistence has deleterious welfare consequences due to pervasive poverty. The author uses fractional integration methods and verify that inflation series are characterized by mean-reversion, finite variance and asymmetry in inflation persistence.

Meanwhile, Duarte and Rua (2007) appraise inflation forecast accuracy over short-term horizon via CPI disaggregated data. The authors adopt a bottom-up approach, aggregating forecasts and later comparing against predictions obtained using the aggregated CPI. The findings reported suggest that, for very short-term inflation forecasting, the bottom-up approach is superior, while, for longer horizons, simpler models fitted with aggregated data tend to display stronger performance.

Finally, an interesting extension related to the optimization of neural networks involve robust methods. Briefly, whereas traditional optimization methods attempt to simply find the minimum of a function over the set of feasible solutions, robust methods take into consideration the uncertainty associated with the inputs of the model to find a solution that may not be theoretically the best, but that may perform decently considering all the possible realizations of the parameters. Indeed, as emphasized by Gabrel, Murat and Thiele (2014), robust optimization encompasses several approaches designed to protect the decision-maker against parameter ambiguity and stochastic uncertainty. Mathematically, given an $\varepsilon > 0$ and a function $f(x)$ to be minimized, robust optimization methods focus on solving the following type of problem:

$$\min_{x} \max_{\|\Delta x\| \leq \varepsilon} f(x + \Delta x) \tag{6.2}$$

That is, they seek to minimize the maximum value attained by the function f given the uncertainty set generated by deviations $\|\Delta x\| \leq \varepsilon$.

Until now, robust optimization has exhibited impressive results in portfolio selection problems and analogous contexts where input uncertainty has a significant influence on the optimal decision. For instance, Bertsimas and Pachamanova (2008) apply these methods to select portfolios in a multiperiod framework under the presence of transaction costs. The authors conclude that robust portfolios are far more resilient to parameter ambiguity and, thus, outperform traditional portfolios in the real world. Further applications across a broad spectrum of domains are provided by Bertsimas, Brown and Caramanis (2011), covering examples belonging to Finance, Statistics, Learning, and Engineering.

In the context of inflation forecasting, given that data is prone to revisions and may suffer from questionable methodologies for their computation, it could potentially improve out-of-sample performance when training neural networks. Besides, as indicated by Keskar et al. (2017), robust optimization could also mitigate the impacts of sharp minima in the out-of-sample loss function and accuracy.

BIBLIOGRAPHY

ADAMOWSKI, J. F. Peak daily water demand forecast modeling using artificial neural networks. *Journal of Water Resources Planning and Management*, v. 134, n. 2, p. 119–128, Mar. 2008. Cited on page 73.

AHMED, N. K. et al. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, v. 29, n. 5, p. 594–621, Sep. 2010. Cited 2 times on pages 28 and 73.

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, v. 46, n. 3, p. 175–185, Aug. 1992. Cited on page 51.

ÁLVAREZ-DÍAZ, M.; GUPTA, R. Forecasting US consumer price index: Does nonlinearity matter? *Applied Economics*, v. 48, n. 46, p. 4462–4475, Mar. 2016. Cited 2 times on pages 28 and 36.

ARUOBA, S. B.; BOCOLA, L.; SCHORFHEIDE, F. Assessing DSGE model nonlinearities. *Journal of Economic Dynamics & Control*, v. 83, n. 10, p. 34–54, Oct. 2017. Cited on page 36.

ATKESON, A.; OHANIAN, L. E. Are Phillips Curves useful for forecasting inflation? *Federal Reserve Bank of Minneapolis Quarterly Review*, Minneapolis, v. 25, n. 1, p. 2–11, Jan./Mar. 2001. Cited 12 times on pages 21, 22, 35, 90, 92, 110, 111, 112, 113, 114, 115, and 119.

ATSALAKIS, G. S.; VALAVANIS, K. P. Surveying stock market forecasting techniques – part II: Soft computing methods. *Expert Systems with Applications*, v. 36, n. 3, p. 5932–5941, Apr. 2009. Cited 4 times on pages 28, 56, 73, and 107.

BAI, J.; NG, S. Forecasting economic time series using targeted predictors. *Journal of Econometrics*, v. 146, n. 2, p. 304–317, Oct. 2008. Cited 2 times on pages 41 and 91.

BAI, J.; NG, S. Boosting diffusion indices. *Journal of Applied Econometrics*, v. 24, n. 4, p. 607–629, Mar. 2009. Cited on page 42.

BAI, S.; TANG, H.; AN, S. Coordinate CNNs and LSTMs to categorize scene images with multi-views and multi-levels of abstraction. *Expert Systems with Applications*, v. 120, n. 3, p. 298–309, Apr. 2019. Cited 2 times on pages 65 and 78.

BAILLIE, R. T.; CHUNG, C.-F.; TIESLAY, M. A. Analysing inflation by the fractionally integrated ARFIMA-GARCH model. *Journal of Applied Econometrics*, v. 11, n. 1, p. 23–40, Jan. 1996. Cited on page 37.

BALL, L.; MAZUMDER, S. Inflation dynamics and the Great Recession. *Brookings Papers on Economic Activity*, v. 42, n. 1, p. 337–405, Jul. 2011. Cited on page 36.

BAO, W.; YUE, J.; RAO, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS ONE*, v. 12, n. 7, p. 203–228, Jul. 2017. Cited 4 times on pages 29, 73, 86, and 132.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 8, p. 1798–1828, Ago. 2013. Cited on page 61.

BERTSIMAS, D.; BROWN, D. B.; CARAMANIS, C. Theory and applications of robust optimization. *SIAM Review*, v. 53, n. 3, p. 464–501, Sep. 2011. Cited on page 134.

BERTSIMAS, D.; PACHAMANOVA, D. Robust multiperiod portfolio management in the presence of transaction costs. *Computer & Operations Research*, v. 35, n. 1, p. 3–17, Jan. 2008. Cited on page 134.

BINNER, J. M. et al. Predictable non-linearities in U.S. inflation. *Economics Letters*, v. 93, n. 3, p. 323–328, Dec. 2006. Cited on page 36.

BISHOP, C. M. *Pattern recognition and machine learning*. 1. ed. Cambridge: Springer, 2006. Cited 2 times on pages 54 and 83.

BLANCHARD, O. The Phillips Curve: Back to the '60s? *American Economic Review*, v. 106, n. 5, p. 31–34, May 2016. Cited on page 35.

BLOOM, N. The impact of uncertainty shocks. *Econometrica*, v. 77, n. 3, p. 623–685, May 2009. Cited on page 27.

BOIVIN, J.; NG, S. Are more data always better for factor analysis? *Journal of Econometrics*, v. 132, n. 1, p. 169–194, May 2006. Cited on page 74.

BOLLERSLEV, T. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, v. 31, n. 3, p. 307–327, Apr. 1986. Cited on page 38.

BREIMAN, L. Bagging predictors. *Machine Learning*, v. 24, n. 1, p. 123–140, Aug. 1996. Cited on page 46.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, Oct. 2001. Cited 2 times on pages 47 and 50.

CAI, M.; LIU, J. Maxout neurons for deep convolutional and LSTM neural networks in speech recognition. *Speech Communication*, v. 77, n. 1, p. 53–64, Mar. 2016. Cited on page 70.

CALVO, G. A. Staggered prices in a utility-maximizing framework. *Journal of Monetary Economics*, v. 12, n. 3, p. 383–398, Sep. 1983. Cited on page 33.

CAO, J.; LI, Z.; LI, J. Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A: Statistical Mechanics and its Applications*, v. 519, n. 4, p. 1509–1531, Apr. 2019. Cited on page 68.

CHANG, Z.; ZHANG, Y.; CHEN, W. Electricity price prediction based on hybrid model of adam optimized LSTM neural network and wavelet transform. *Energy*, v. 187, n. 1, p. 1–12, Nov. 2019. Cited on page 132.

CHEN, G. et al. Multiple random forests modelling for urban water consumption forecasting. *Water Resources Management*, v. 31, n. 1, p. 4715–4729, Sep. 2017. Cited on page 50.

CHIPMAN, H. A.; GEORGE, E. I.; MCCULLOCH, R. E. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, v. 4, n. 1, p. 266–298, Jan. 2010. Cited 2 times on pages 50 and 51.

CHO, K. et al. Understanding batch normalization. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEM, 31., 2018, Montreal, Canada. *Proceedings...* [S.l.]: Curran Associates, Inc., 2014. p. 7694–7705. Cited on page 85.

CHONG, E.; HAN, C.; PARK, F. C. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, v. 83, n. 1, p. 187–205, Oct. 2017. Cited 2 times on pages 97 and 107.

CHOUDHARY, M. A.; HAIDER, A. Neural network models for inflation forecasting: An appraisal. *Applied Economics*, v. 44, n. 20, p. 2631–2635, Oct. 2012. Cited 2 times on pages 28 and 71.

CLARK, T. E.; DOH, T. Evaluating alternative models of trend inflation. *International Journal of Forecasting*, v. 30, n. 3, p. 426–448, Jul./Sep. 2014. Cited on page 37.

COIBION, O.; GORODNICHENKO, Y. Is the Phillips Curve alive and well after all? Inflation expectations and the missing disinflation. *American Economic Journal: Macroeconomics*, v. 7, n. 1, p. 197–232, Jan. 2015. Cited on page 36.

COLEMAN, S. Inflation persistence in the franc zone: Evidence from disaggregated prices. *Journal of Macroeconomics*, v. 32, n. 1, p. 426–442, Mar. 2010. Cited on page 134.

COLLINS, M.; SCHAPIRE, R. E.; SINGER, Y. Logistic regression, AdaBoost, and Bregman distances. *Machine Learning*, v. 48, n. 1, p. 253–285, Jul. 2002. Cited on page 46.

COLOGNI, A.; MANERA, M. Oil prices, inflation and interest rates in a structural cointegrated VAR model for the G-7 countries. *Energy Economics*, v. 30, n. 3, p. 856–888, May 2008. Cited 2 times on pages 39 and 92.

CORREA, A. S.; MINELLA, A. Nonlinear mechanisms of the exchange rate pass-through: A Phillips curve model with threshold for Brazil. *Revista Brasileira de Economia*, Rio de Janeiro, v. 64, n. 3, p. 231–243, Jul./Sep. 2010. Cited on page 36.

COURBARIAUX, M.; BENGIO, Y.; DAVID, J.-P. BinaryConnect: Training deep neural networks with binary weights during propagations. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 28., 2015, Montreal, Canada. *Proceedings...* [S.l.]: MIT Press, 2015. p. 3123–3131. Cited on page 133.

DALY, M. C.; HOBIJN, B. Downward nominal wage rigidities bend the Phillips Curve. *Journal of Money, Credit and Banking*, v. 46, n. 2, p. 51–93, Oct. 2014. Cited 2 times on pages 27 and 36.

DASH, N. B. et al. Hybrid neural modeling for groundwater level prediction. *Neural Computing and Applications*, v. 19, n. 8, p. 1251–1263, Nov. 2010. Cited on page 78.

DELALLEAU, O.; BENGIO, Y. Shallow vs. deep sum-product networks. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 24., 2011, Granada, Spain. *Proceedings...* Red Hook, NY: NIPS, 2011. p. 666–674. Cited on page 53.

DENNIS, R. *The frequency of price adjustment and New Keynesian business cycle dynamics*. San Francisco, 2006. (Federal Reserve Bank of San Francisco Working Paper Series). Cited on page 34.

DHILLON, A.; VERMA, G. K. Convolutional neural network: A review of models, methodologies and application to object detection. *Progress in Artificial Intelligence*, v. 9, n. 2, p. 85–112, Jun. 2020. Cited on page 64.

DOERSCH, C. *Tutorial on Variational Autoencoders*. 2016. Cited 2 times on pages 74 and 75.

DONG, G. et al. A review of the autoencoder and its variants: A comparative perspective from target recognition in synthetic-aperture radar images. *IEEE Geoscience and Remote Sensing Magazine*, v. 6, n. 3, p. 51–93, Oct. 2018. Cited 3 times on pages 17, 62, and 133.

DOZAT, T. Incorporating Nesterov momentum into Adam. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 4., 2016, San Juan, Puerto Rico. *Proceedings...* San Juan, Puerto Rico: ICLR, 2016. p. 1–4. Cited on page 98.

DUARTE, C.; RUA, A. Forecasting inflation through a bottom-up approach: How bottom is bottom? *Economic Modelling*, v. 24, n. 6, p. 941–953, Nov. 2007. Cited on page 134.

ELDAN, R.; SHAMIR, O. The power of depth for feedforward neural networks. In: CONFERENCE ON LEARNING THEORY, 29., 2016, Columbia University, New York. *Proceedings of Machine Learning Research*. New York: PMLR, 2016. p. 907–940. Cited on page 53.

ESSIEN, A.; GIANNETTI, C. A deep learning framework for univariate time series prediction using convolutional LSTM stacked autoencoders. In: IEEE INTERNATIONAL SYMPOSIUIM ON INNOVATIONS IN INTELLIGENT SYSTEMS AND APPLICATIONS, 29., 2016, Sofia, Bulgaria. *Proceedings...* Sofia: IEEE, 2019. p. 1–6. Cited 4 times on pages 29, 66, 68, and 73.

FAUST, J.; WRIGHT, J. H. Forecasting inflation. In: ELLIOTT, G.; TIMMERMANN, A. *Handbook of Economic Forecasting*. 1. ed. New York: Elsevier, 2013. p. 2–56. Cited on page 27.

FISCHER, T.; KRAUSS, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, v. 270, n. 2, p. 654–669, Oct. 2018. Cited 5 times on pages 17, 29, 68, 70, and 86.

FRIEDMAN, J. H. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, v. 38, n. 4, p. 367–378, Feb. 2002. Cited 2 times on pages 45 and 46.

GABREL, V.; MURAT, C.; THIELE, A. Recent advances in robust optimization: An overview. *European Journal of Operational Research*, v. 235, n. 3, p. 471–483, Jun. 2014. Cited on page 134.

GALESHCHUK, S. Neural networks performance in exchange rate prediction. *Neurocomputing*, v. 172, n. 1, p. 446–452, Jan. 2016. Cited on page 73.

GALI, J. *Monetary Policy, Inflation, and the Business Cycle*. 2. ed. New Jersey: Princeton University Press, 2015. Cited on page 33.

GALI, J.; GERTLER, M. Inflation dynamics: A structural econometric analysis. *Journal of Monetary Economics*, v. 44, n. 2, p. 195–222, Oct. 1999. Cited 2 times on pages 33 and 35.

GALLEGATI, M. Wavelet analysis of stock returns and aggregate economic activity. *Computational Statistics & Data Analysis*, v. 52, n. 6, p. 3061–3074, Feb. 2008. Cited on page 132.

GARCIA, M. G. P.; MEDEIROS, M. C.; VASCONCELOS, G. F. R. Real-time inflation forecasting with high-dimensional models: The case of Brazil. *International Journal of Forecasting*, v. 33, n. 3, p. 679–693, Jul./Sep. 2017. Cited 2 times on pages 28 and 71.

GELMAN, A. et al. *Bayesian Data Science.* 2. ed. Florida: CRC Press, 2004. Cited on page 92.

GERS, F.; SCHMIDHUBER, J. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, v. 12, n. 6, p. 1333–1340, Nov. 2001. Cited on page 68.

GERS, F.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with LSTM. *Neural Computation*, v. 12, n. 10, p. 2451–2471, Out. 2000. Cited 3 times on pages 68, 69, and 86.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS, 15., 2011, Columbia University, New York. *Proceeding of Machine Learning Research.* New York: PMLR, 2011. p. 315–323. Cited 2 times on pages 56 and 67.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning.* 1. ed. Massachusetts: The MIT Press, 2016. Cited 19 times on pages 17, 28, 29, 47, 54, 59, 61, 62, 63, 64, 66, 67, 75, 76, 83, 85, 86, 94, and 100.

GOODFELLOW, I. et al. Generative adversarial nets. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 27., 2014, Cambridge, USA. *Proceedings...* [S.l.]: Association for Computational Linguistics, 2014. p. 2672–2680. Cited on page 133.

GRAUWE, P. D.; JI, Y. Inflation targets and the zero lower bound in a behavioral macroeconomic model. *Economica*, v. 86, n. 342, p. 262–299, Apr. 2019. Cited on page 27.

GREFF, K. et al. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, v. 28, n. 10, p. 2222–2232, Oct. 2017. Cited on page 70.

GU, S.; KELLY, B.; XIU, D. Autoencoder asset pricing models. *Journal of Econometrics*, Jul. 2020. Available at: <https://doi.org/10.1016/j.jeconom.2020.07.009>. Last access: 04 Jan. 2021. Cited 3 times on pages 30, 120, and 131.

GUPTA, R.; KABUNDI, A. A large factor model for forecasting macroeconomic variables in South Africa. *International Journal of Forecasting*, v. 27, n. 4, p. 1, Oct./Dec. 2011. Cited on page 42.

HAMILTON, J. D. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, v. 57, n. 2, p. 357–384, Mar. 1989. Cited on page 40.

HANS, C. Bayesian lasso regression. *Biometrika*, v. 96, n. 4, p. 835–845, Sep. 2009. Cited on page 43.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning*. 2. ed. California: Springer, 2008. Cited 12 times on pages 17, 37, 42, 43, 44, 45, 46, 47, 48, 49, 51, and 52.

HAYKIN, S. *Neural networks*: A comprehensive foundation. 2. ed. Ontario: Prentice Hall, 2004. Cited 2 times on pages 54 and 58.

HINTON, G. E.; OSINDERO, S.; TEH, Y. A fast learning algorithm for deep belief nets. *Neural Computation*, v. 18, n. 7, p. 1527–1554, Jul. 2006. Cited on page 53.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 1, p. 1735–1780, Nov. 1997. Cited on page 68.

HOERL, A. E.; KENNARD, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, v. 12, n. 1, p. 55–67, Feb. 1970. Cited on page 44.

HOFFER, E.; HUBARA, I.; SOUDRY, D. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. *Advances in Neural Information Processing Systems*, v. 30, n. 1, p. 1729–1739, Feb. 2017. Cited on page 100.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v. 2, n. 5, p. 359–366, Sep./Oct. 1989. Cited on page 58.

HOSKING, J. R. M. Fractional differencing. *Biometrika*, v. 68, n. 1981, p. 165–176, Apr. 1981. Cited on page 37.

HSIEH, T.-J.; HSIAO, H.-F.; YEH, W.-C. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, v. 11, n. 2, p. 2510–2525, Mar. 2011. Cited 2 times on pages 66 and 132.

HÜSKEN, M.; STAGGE, P. Recurrent neural networks for time series classification. *Neurocomputing*, v. 50, n. 1, p. 223–235, Jan. 2003. Cited on page 65.

INOUE, A.; KILIAN, L. How useful is bagging in forecasting economic time series?: A case study of U.S. consumer price inflation. *Journal of the American Statistical Association*, v. 103, n. 482, p. 511–522, Jun. 2008. Cited on page 47.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 32., 2015, Lille, France. *Proceedings...* Lille, France: JMLR, 2015. p. 448–456. Cited on page 85.

JAMES, G. et al. *An Introduction to Statistical Learning with Application in R*. 1. ed. New York: Springer New York, 2013. Cited 4 times on pages 37, 42, 95, and 108.

JOZEFOWICZ, R.; ZAREMBA, W.; SUTSKEVER, I. An empirical exploration of recurrent network architectures. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2015. *Proceedings...* New York: ACM, 2015. p. 2342–2350. Cited on page 70.

KESKAR, N. S. et al. On large-batch training for deep learning: Generalization gap and sharp minima. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 5., 2017, Toulon, France. *Proceedings...* Toulon, France: ICLR, 2017. p. 1–13. Cited 4 times on pages 18, 99, 100, and 135.

KHADAROO, A. J. A threshold in inflation dynamics: Evidence from emerging countries. *Applied Economics*, v. 37, n. 6, p. 719–723, Jun. 2005. Cited on page 41.

KHASHEI, M.; BIJARI, M. A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Applied Soft Computing*, v. 11, n. 2, p. 2664–2675, Mar. 2011. Cited on page 78.

KIM, C. et al. Artificial neural networks for non-stationary time series. *Neurocomputing*, v. 61, n. 8, p. 439–447, Oct. 2004. Cited on page 73.

KIM, C.-J. Unobserved-component time series models with Markov-switching heteroscedasticity: Changes in regime and the link between inflation rates and inflation uncertainty. *Journal of Business & Economic Statistics*, v. 11, n. 3, p. 341–349, Jul. 1993. Cited on page 120.

KIM, H. H.; SWANSON, N. R. Forecasting financial and macroeconomic variables using data reduction methods: New empirical evidence. *Journal of Econometrics*, v. 178, n. 2, p. 352–367, Jan. 2014. Cited on page 45.

KIM, H. Y.; WON, C. H. Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. *Expert Systems with Applications*, v. 103, n. 1, p. 25–37, Aug. 2018. Cited 2 times on pages 70 and 73.

KIM, J. et al. Deep neural network with weight sparsity control and pre-training extracts hierarchical features and enhances classification performance: Evidence from whole-brain resting-state functional connectivity paters of schizophrenia. *NeuroImage*, v. 124, n. 1, p. 127–146, Jan. 2016. Cited on page 28.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 3., 2015, San Diego, California. *Proceedings...* Ithaca, New York: ICLR, 2015. p. 1–13. Cited on page 98.

KINGMA, D. P.; WEILLING, M. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, v. 12, n. 4, p. 307–392, Apr. 2019. Cited 2 times on pages 75 and 77.

KLAMBAUER, G. et al. Self-normalizing neural networks. In: CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 31., 2017, Long Beach, California. *Proceedings...* Long Beach: NIPS, 2017. p. 315–323. Cited 2 times on pages 57 and 82.

KONTONIKAS, A. Inflation and inflation uncertainty in the United Kingdom, evidence from GARCH modelling. *Economic Modelling*, v. 21, n. 3, p. 525–543, May 2004. Cited on page 38.

KRISTJANPOLLER, W.; MINUTOLO, M. C. Gold price volatility: A forecasting approach using the Artificial Neural Network-GARCH model. *Expert Systems with Applications*, v. 42, n. 20, p. 7245–7251, Nov. 2015. Cited on page 78.

KUHN, M.; JOHNSON, K. *Applied predictive modeling*. 1. ed. New York: Springer, 2013. Cited on page 95.

KUMAR, A.; ORRENIUS, P. M. A closer look at the Phillips curve using state-level data. *Journal of Macroeconomics*, v. 47, n. 3, p. 84–102, Mar. 2016. Cited on page 27.

LÄNGKVIST, M.; KARLSSON, L.; LOUTFI, A. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, v. 42, n. 2, p. 11–24, Jun. 2017. Cited 2 times on pages 29 and 73.

LECUN, Y. et al. Convolutional networks and applications in vision. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 1., 2010, Paris, France. *Proceedings...* Paris, France: IEEE, 2010. p. 1–4. Cited on page 65.

LEE, S. W.; KIM, H. Y. Stock market forecasting with super-high dimensional time-series data using ConvLSTM, trend sampling, and specialized data augmentation. *Expert Systems with Applications*, v. 161, n. 1, p. 1–20, Dec. 2020. Cited on page 78.

LEROUGE, J. et al. IODA: An input/output deep architecture for image labeling. *Pattern Recognition*, v. 48, n. 9, p. 2847–2858, Sep. 2015. Cited on page 28.

LI, J.; CHEN, W. Forecasting macroeconomic time series: LASSO-based approaches and their forecast combinations with dynamic factor models. *International Journal of Forecasting*, v. 30, n. 4, p. 996–1015, Oct.-Dec. 2014. Cited on page 44.

LI, X.; WU, X. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 1., 2015, Brisbane, QLD. *Proceedings...* Brisbane: IEEE, 2015. p. 4520–4524. Cited 2 times on pages 28 and 70.

LIU, G.; GUO, J. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing*, v. 337, n. 1, p. 325–338, Apr. 2019. Cited on page 70.

LIU, W. et al. A survey of deep neural network architectures and their applications. *Neurocomputing*, v. 234, n. 1, p. 11–26, Apr. 2017. Cited 2 times on pages 28 and 73.

LU, C. et al. Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification. *Signal Processing*, v. 130, n. 1, p. 377–388, Jan. 2017. Cited on page 61.

LÜTKEPOHL, H. *New introduction to multiple time series analysis*. 1. ed. Berlin: Springer, 2005. Cited on page 38.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 30., 2013, Atlanta, GA. *Proceedings...* Atlanta, GA: ICML, 2013. p. 1–13. Cited on page 57.

MAVROEIDIS, S. Identification issues in forward-looking models estimated by GMM, with an application to the Phillips Curve. *Journal of Money, Credit and Banking*, v. 37, n. 3, p. 421–448, Jun. 2005. Cited on page 35.

MAVROEIDIS, S.; PLAGBORG-MØLLER, M.; STOCK, J. H. Empirical evidence on inflation expectations in the New Keynesian Phillips Curve. *Journal of Economic Literature*, v. 52, n. 1, p. 124–188, Mar. 2014. Cited on page 33.

MCADAM, P.; MCNELIS, P. Forecasting inflation with thick models and neural networks. *Economic Modelling*, v. 22, n. 5, p. 848–867, Sep. 2005. Cited 2 times on pages 28 and 70.

MCCRACKEN, M.; NG, S. FRED-MD: A monthly database for macroeconomic research. *Journal of Business and Economic Statistics*, v. 34, n. 4, p. 574–589, Sep. 2016. Cited 12 times on pages 23, 24, 27, 74, 93, 104, 133, 149, 150, 151, 152, and 153.

MEDEIROS, M. C. et al. Forecasting inflation in a data-rich environment: The benefits of machine learning methods. *Journal of Business and Economic Statistics*, Aug. 2019. Available at: <https://doi.org/10.1080/07350015.2019.1637745>. Last access: 27 Jun. 2020. Cited 10 times on pages 15, 27, 28, 33, 70, 71, 91, 93, 119, and 120.

MESSINA, R.; LOURADOUR, J. Segmentation-free handwritten Chinese text recognition with LSTM-RNN. In: INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION, 13., 2015, Tunis, Tunisia. *Proceedings...* Tunis: IEEE, 2015. p. 171–175. Cited on page 70.

MONACELLI, T.; SALA, L. The international dimension of inflation: Evidence from disaggregated consumer price data. *Journal of Money, Credit and Banking*, v. 41, n. 1, p. 101–120, Jan. 2009. Cited on page 134.

MONACHE, D. D.; PETRELLA, I. Adaptive models and heavy tails with an application to inflation forecasting. *International Journal of Forecasting*, v. 33, n. 2, p. 482–501, Apr./Jun. 2017. Cited 2 times on pages 37 and 104.

MURRAY, N.; PERRONNIN, F. Generalized max pooling. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 27., 2014, Columbus, Ohio. *Proceedings...* Columbus, Ohio: IEEE, 2014. p. 2473–2480. Cited on page 86.

NIU, Z. et al. Wind power forecasting using attention-based gated recurrent unit network. *Energy*, v. 196, n. 1, p. 1–17, Apr. 2020. Cited on page 68.

NTI, K. O.; ADEKOYA, A.; WEYORI, B. Random forest based feature selection of macroeconomic variables for stock market prediction. *American Journal of Applied Sciences*, v. 16, n. 7, p. 200–212, Sep. 2019. Cited on page 50.

O'REILLY, R. C.; FRANK, M. J. Making working memory work: A computational model of learning in the prefontral cortex and basal ganglia. *Neural Computation*, v. 18, n. 2, p. 283–328, Feb. 2006. Cited on page 70.

PALANGI, H.; WARD, R.; DENG, L. Distributed compressive sensing: A deep learning approach. *IEEE Transactions on Signal Processing*, v. 64, n. 17, p. 4504–4518, Sep. 2016. Cited on page 67.

PARK, T.; CASELLA, G. The Bayesian Lasso. *Journal of the American Statistical Association*, v. 103, n. 482, p. 681–686, Jun. 2008. Cited 3 times on pages 43, 44, and 91.

PERCIVAL, D. B.; WALDEN, A. T. *Wavelet Methods for Time Series Analysis.* 2. ed. Cambridge: Cambridge University Press, 2006. Cited on page 132.

PEREIRA, J.; SILVEIRA, M. Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS, 17., 2018, Orlando, FL. *Proceedings...* Orlando, FL: IEEE, 2018. p. 1–13. Cited on page 74.

PRÜSER, J. Forecasting with many predictors using bayesian additive regression trees. *Journal of Forecasting*, v. 38, n. 7, p. 621–631, Mar. 2019. Cited on page 51.

RAMACHANDRAN, P.; ZOPH, B.; LE, V. Searching for activation functions. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 6., 2018, Vancouver, BC. *Proceedings...* Vancouver, BC: ICLR, 2018. p. 1–13. Cited on page 132.

RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Networks*, v. 29, n. 9, p. 2352–2449, Sep. 2017. Cited on page 63.

ROUX, N. L.; BENGIO, Y. Deep belief networks are compact universal approximators. *Neural Computation*, v. 22, n. 8, p. 2192–2207, Aug. 2010. Cited on page 53.

RUDD, J.; WHELAN, K. Modeling inflation dynamics: A critical review of recent research. *Journal of Money, Credit and Banking*, v. 39, n. 1, p. 155–170, Feb. 2007. Cited on page 27.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 1, p. 533–536, Oct. 1986. Cited on page 65.

SANTURKAR, S. et al. How does batch normalization help optimization? In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING, 32., 2018, New York, United States. *Proceedings...* New York, United States: Curran Associates, 2018. p. 2488–2498. Cited on page 85.

SCHÄFER, A. M.; ZIMMERMAN, H. G. Recurrent neural networks are universal approximators. *International Journal of Neural Systems*, v. 17, n. 4, p. 253–263, Jun. 2007. Cited on page 66.

SCHALING, E. The nonlinear Phillips Curve and inflation forecast targeting: Symmetric versus asymmetric monetary policy rules. *Journal of Money, Credit and Banking*, v. 36, n. 3, p. 361–386, Jun. 2004. Cited on page 35.

SCHERER, D.; MÜLLER, A. Evaluation of pooling operations in convolutional architectures for object recognition. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS, 20., 2010, Thessaloniki, Greece. *Proceedings...* Thessaloniki, Greece: ICANN, 2010. p. 92–101. Cited on page 86.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, v. 61, n. 1, p. 85–117, Jan. 2015. Cited 2 times on pages 53 and 67.

SEZER, O. B.; OZBAYOGLU, A. M. Algorithmic financial trading with deep convolutional networks: Time series to image conversion approach. *Applied Soft Computing*, v. 70, n. 1, p. 525–538, Sep. 2018. Cited 3 times on pages 64, 78, and 82.

SHEN, G. et al. Deep learning with gated recurrent unit networks for financial predictions. *Procedia Computer Science*, v. 131, n. 1, p. 895–903, Apr. 2018. Cited 2 times on pages 67 and 68.

SHI, X. et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 28., 2015, Cambridge, MA. *Proceedings...* Cambridge: NIPS, 2015. p. 802–810. Cited 5 times on pages 29, 73, 77, 78, and 82.

SIBI, P.; JONES, S. A.; SIDDARTH, P. Analysis of different activation functions using back proparagion neural networks. *Journal of Theoretical and Applied Information Technology*, v. 43, n. 3, p. 1264–1268, Jan. 2013. Cited on page 56.

SILVERMAN, B. *Density Estimation for Statistics and Data Analysis*. 1. ed. London: Chapman and Hall, 1986. Cited on page 108.

SONG, Y. et al. An efficient instance selection algorithm for k nearest neighbor regression. *Neurocomputing*, v. 251, n. 1, p. 26–34, Aug. 2017. Cited on page 51.

SOYDANER, D. A comparison of optimization algorithms for deep learning. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 34, n. 13, p. 1264–1268, Apr. 2020. Cited on page 98.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 6, p. 1929–1958, Jun. 2014. Cited 3 times on pages 18, 83, and 84.

STOCK, J. H.; WATSON, M. W. Forecasting inflation. *Journal of Monetary Economics*, v. 44, n. 2, p. 293–335, Oct. 1999. Cited on page 71.

STOCK, J. H.; WATSON, M. W. Why has US inflation become harder to forecast? *Journal of Money, Credit and Banking*, v. 39, n. 1, p. 3–33, Feb. 2007. Cited on page 71.

STOCK, J. H.; WATSON, M. W. Dynamic factor models, factor-augmented vector autoregressions, and structural vector autoregressions in macroeconomics. In: J. B. TAYLOR AND H. UHLIG. *Handbook of Macroecoomics*. 1. ed. New York: Elsevier, 2016. v. 2, p. 415–525. Cited on page 41.

TAKAHASHI, S.; CHEN, Y.; TANAKA-ISHII, K. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and Its Applications*, v. 527, n. 1, p. 1212–1261, Aug. 2019. Cited on page 133.

TANG, Y.; ELIASMITH, C. Deep networks for robust visual recognition. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 27., 2010, Haifa, Israel. *Proceedings...* Haifa: NIPS, 2010. p. 21–24. Cited on page 83.

TIBSHIRANI, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, v. 58, n. 1, p. 267–288, Jan. 1996. Cited on page 42.

TIPPING, M. E. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, v. 1, n. 1, p. 211–244, Jan. 2001. Cited on page 44.

TKÁČ, M.; VERNER, R. Artificial neural networks in business: Two decades of research. *Applied Soft Computing*, v. 38, n. 1, p. 788–804, Jan. 2016. Cited on page 29.

TSAY, R. S. *Analysis of Financial Time Series*. 3. ed. New Jersey: Wiley, 2010. Cited 4 times on pages 37, 38, 39, and 40.

ÜLKE, V. et al. A comparison of time series and machine learning models for inflation forecasting: Empirical evidence from the USA. *Neural Computing and Applications*, v. 30, n. 5, p. 1519–1527, Sep. 2018. Cited 2 times on pages 28 and 37.

VASCONCELOS, G. F. R. *Forecasting in high-dimension*: Inflation and other economic variables. 169 p. Tese (Doutorado em Engenharia Elétrica) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2018. Cited 2 times on pages 48 and 49.

VINCENT, P. et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, v. 11, n. 1, p. 3371–3408, Dec. 2010. Cited 3 times on pages 59, 60, and 61.

WANG, K.; QI, X.; LIU, H. Photovoltaic power forecasting based LSTM-Convolutional network. *Energy*, v. 189, n. 1, p. 1–11, Dec. 2019. Cited 2 times on pages 29 and 73.

WANG, Y.; YAO, H.; ZHAO, S. Auto-encoder based dimensionality reduction. *Neurocomputing*, v. 184, n. 1, p. 232–242, Apr. 2016. Cited 4 times on pages 29, 30, 61, and 62.

YAMASHITA, R. et al. Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, v. 9, n. 1, p. 611–629, Jun. 2018. Cited on page 65.

YU, Y. et al. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, v. 31, n. 7, p. 1235–1270, Jul. 2019. Cited 3 times on pages 68, 87, and 133.

ZHANG, B.; WU, J.-L.; CHANG, P.-C. A multiple time series-based recurrent neural network for short-term load forecasting. *Soft Computing*, v. 22, n. 12, p. 4099–4112, Jun. 2018. Cited on page 66.

ZHANG, L. Modeling the Phillips curve in China: A nonlinear perspective. *Macroeconomic Dynamics*, v. 21, n. 2, p. 439–461, Mar. 2017. Cited on page 27.

ZHAO, B. et al. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, v. 28, n. 1, p. 162–169, Feb. 2017. Cited on page 65.

ZHOU, G. et al. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, v. 13, n. 3, p. 226–234, Jun. 2016. Cited on page 68.

ZHOU, Y.; CHELLAPPA, R. Computation of optical flow using a neural network. In: INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, 1., 1988, San Diego, CA. *Proceedings...* San Diego, CA: IEEE, 1988. p. 71–78. Cited on page 86.

APPENDIX A – FULL DESCRIPTION OF THE DATASET

The next tables contain the full details of the dataset employed to fit the model developed and its benchmarks.

Table 11 – Macroeconomic time series used to fit the model developed for inflation forecasting and its benchmarks. "Transf." refers to the transformation applied to the data according to the recommendations given by McCracken and Ng (2016), which are compiled in Table 13.

Group	Transf.	FRED Code	Description
OI	5	RPI	Real Personal Income (RPI)
OI	5	W875RX1	RPI ex Transfer Receipts
OI	5	INDPRO	Industrial Production (IP) Index
OI	5	IPFPNSS	IP - Final Products
OI	5	IPFINAL	IP - Final Products (Market Group)
OI	5	IPCONGD	IP - Consumer Goods
OI	5	IPDCONGD	IP - Durable Goods
OI	5	IPNCONGD	IP - Nondurable Consumer Goods
OI	5	IPBUSEQ	IP - Business Equipment
OI	5	IPMAT	IP - Materials
OI	5	IPDMAT	IP - Durable Materials
OI	5	IPNMAT	IP - Nondurable Materials
OI	5	IPMANSICS	IP - Manufacturing (ISC)
OI	5	IPB51222s	IP - Residential Utilities
OI	5	IPFUELS	IP - Fuels
OI	1	NAPMPI	ISM Manufacturing - Prod. Index
OI	2	CUMFNS	Capacity Utilization - Manufacturing
LM	2	HWI	Help-Wanted Index for US
LM	2	HWIURATIO	Ratio of Help Wanted / No. Unemp.
LM	5	CLF16OV	Civilian Labor Force
LM	5	CE16OV	Civilian Employment
LM	2	UNRATE	Civilian Unemployment Rate
LM	2	UEMPMEAN	Avg. Duration of Unemp. (Weeks)
LM	5	UEMPLT5	Civ. Unemp. - Less Than 5 Weeks
LM	5	UEMP5TO14	Civ. Unemp. for 5-14 Weeks
LM	5	UEMP15OV	Civ. Unemp. for 15 Weeks and Over
LM	5	UEMP15T26	Civ. Unemp. for 15-26 Weeks
LM	5	UEMP27OV	Civ. Unemp. for 27 Weeks and Over
LM	5	CLAIMSx	Initial Claims
LM	5	PAYEMS	All Emp.: Total nonfarm

Table 11 – Macroeconomic time series used to fit the model developed for inflation forecasting and its benchmarks. “Transf.” refers to the transformation applied to the data according to the recommendations given by McCracken and Ng (2016), which are compiled in Table 13.

Group	**Transf.**	**FRED Code**	**Description**
LM	5	USGOOD	All Emp.: Goods-Producing Industries
LM	5	CES1021000001	All Emp.: Mining and Logging: Mining
LM	5	USCONS	All Emp.: Construction
LM	5	MANEMP	All Emp.: Manufacturing
LM	5	DMANEMP	All Emp.: Durable goods
LM	5	NDMANEMP	All Emp.: Nondurable goods
LM	5	SRVPRD	All Emp.: Service-Providing Ind.
LM	5	USTPU	All Emp.: Trade, Transp. and Utilities
LM	5	USWTRADE	All Emp.: Wholesale Trade
LM	5	USTRADE	All Emp.: Retail Trade
LM	5	USFIRE	All Emp.: Financial Activities
LM	5	USGOVT	All Emp.: Government
LM	1	CES0600000007	Avg Weekly Hours: Goods-Producing
LM	2	AWOTMAN	Avg Weekly Overtime Hours: Manuf.
LM	1	AWHMAN	Avg Weekly Hours: Manufacturing
LM	1	NAPMEI	ISM Manuf.: Employment Index
LM	6	CES0600000008	Avg Hourly Earn.: Goods-Producing
LM	6	CES2000000008	Avg Hourly Earnings: Construction
LM	6	CES3000000008	Avg Hourly Earnings: Manufacturing
H	4	HOUST	Housing Starts: New Priv. Owned
H	4	HOUSTNE	Housing Starts, Northeast
H	4	HOUSTMW	Housing Starts, Midwest
H	4	HOUSTS	Housing Starts, South
H	4	HOUSTW	Housing Starts, West
H	4	PERMIT	New Private Housing Permits (SAAR)
H	4	PERMITNE	New Priv. Housing Perm., NE (SAAR)
H	4	PERMITMW	New Priv. Housing Perm., MW (SAAR)
H	4	PERMITS	New Priv. Housing Perm., S (SAAR)
H	4	PERMITW	New Priv. Housing Perm., W (SAAR)
COI	5	DPCERA3M086SBEA	Real personal cons. expenditures
COI	5	CMRMTSPLx	Real Manufacturing and Trade Ind.
COI	5	RETAILx	Retail and Food Services Sales
COI	1	NAPM	ISM : PMI Composite Index
COI	1	NAPMNOI	ISM : New Orders Index

Table 11 – Macroeconomic time series used to fit the model developed for inflation forecasting and its benchmarks. "Transf." refers to the transformation applied to the data according to the recommendations given by McCracken and Ng (2016), which are compiled in Table 13.

Group	Transf.	FRED Code	Description
COI	1	NAPMSDI	ISM : Supplier Deliveries Index
COI	1	NAPMII	ISM : Inventories Index
COI	5	AMDMNOx	New Orders for Durable Goods
COI	5	ANDENOx	New Orders for Nondefense Capital
COI	5	AMDMUOx	Unfilled Orders for Durable Goods
COI	5	BUSINVx	Total Business Inventories
COI	2	ISRATIOx	Total Business: Inventories to Sales
COI	2	UMCSENTx	Consumer Sentiment Index
MC	6	M1SL	M1 Money Stock
MC	6	M2SL	M2 Money Stock
MC	5	M2REAL	Real M2 Money Stock
MC	6	AMBSL	St. Louis Adjusted Monetary Base
MC	6	TOTRESNS	Total Reserves of Depository Inst.
MC	7	NONBORRES	Reserves Of Depository Institutions
MC	6	BUSLOANS	Commercial and Industrial Loans
MC	6	REALLN	Real Estate Loans at All Commercial
MC	6	NONREVSL	Total Nonrevolving Credit
MC	2	CONSPI	Nonrevolving Cons. Credit to Pers. Inc.
MC	6	MZMSL	MZM Money Stock
MC	6	DTCOLNVHFNM	Consumer Motor Vehicle Loans Out.
MC	6	DTCTHFNM	Total Consumer Loans and Leases Out.
MC	6	INVEST	Securities in Bank Credit at All
INTFX	2	FEDFUNDS	Effective Federal Funds Rate
INTFX	2	CP3Mx	3-Month AA Fin. Comm. Paper Rate
INTFX	2	TB3MS	3-Month Treasury Bill:
INTFX	2	TB6MS	6-Month Treasury Bill:
INTFX	2	GS1	1-Year Treasury Rate
INTFX	2	GS5	5-Year Treasury Rate
INTFX	2	GS10	10-Year Treasury Rate
INTFX	2	AAA	Moody's Seas. Aaa Corp. Bond Yield
INTFX	2	BAA	Moody's Seas. Baa Corp. Bond Yield
INTFX	1	COMPAPFFx	3M Comm. Paper Minus FEDFUNDS
INTFX	1	TB3SMFFM	3-Month Treasury C Minus FEDFUNDS
INTFX	1	TB6SMFFM	6-Month Treasury C Minus FEDFUNDS

Table 11 – Macroeconomic time series used to fit the model developed for inflation forecasting and its benchmarks. "Transf." refers to the transformation applied to the data according to the recommendations given by McCracken and Ng (2016), which are compiled in Table 13.

Group	Transf.	FRED Code	Description
INTFX	1	T1YFFM	1-Year Treasury C Minus FEDFUNDS
INTFX	1	T5YFFM	5-Year Treasury C Minus FEDFUNDS
INTFX	1	T10YFFM	10-Year Treasury C Minus FEDFUNDS
INTFX	1	AAAFFM	Moody's Aaa Corp. Minus FEDFUNDS
INTFX	1	BAAFFM	Moody's Baa Corp. Minus FEDFUNDS
INTFX	5	TWEXAFEGSMTHx	Trade Weighted U.S. Dollar Index
INTFX	5	EXSZUSx	Switzerland / U.S. Foreign Exch. Rate
INTFX	5	EXJPUSx	Japan / U.S. Foreign Exchange Rate
INTFX	5	EXUSUKx	U.S. / U.K. Foreign Exchange Rate
INTFX	5	EXCAUSx	Canada / U.S. Foreign Exchange Rate
P	6	PPIFGS	PPI: Finished Goods
P	6	PPIFCG	PPI: Finished Consumer Goods
P	6	PPIITM	PPI: Intermediate Materials
P	6	PPICRM	PPI: Crude Materials
P	6	OILPRICEx	Crude Oil spliced WTI and Cushing
P	6	PPICMM	PPI: Metals: nonferrous
P	1	NAPMPRI	ISM Manufacturing: Prices Index
P	6	CPIAUCSL	CPI : All Items
P	6	CPIAPPSL	CPI : Apparel
P	6	CPITRNSL	CPI : Transportation
P	6	CPIMEDSL	CPI : Medical Care
P	6	CUSR0000SAC	CPI : Commodities
P	6	CUUR0000SAD	CPI : Durables
P	6	CUSR0000SAS	CPI : Services
P	6	CPIULFSL	CPI : All Items Less Food
P	6	CUUR0000SA0L2	CPI : All items less shelter
P	6	CUSR0000SA0L5	CPI : All items less medical care
P	6	PCEPI	Personal Cons. Expend.: Chain Index
P	6	DDURRG3M086SBEA	Personal Cons. Exp: Durable goods
P	6	DNDGRG3M086SBEA	Personal Cons. Exp: Nondurable goods
P	6	DSERRG3M086SBEA	Personal Cons. Exp: Services
SM	5	S&P 500	S&P's Price Index: Composite
SM	5	S&P: indust	S&P's Price Index: Industrials
SM	2	S&P div yield	Dividend yield S&P's 500

Table 11 – Macroeconomic time series used to fit the model developed for inflation forecasting and its benchmarks. “Transf.” refers to the transformation applied to the data according to the recommendations given by McCracken and Ng (2016), which are compiled in Table 13.

Group	**Transf.**	**FRED Code**	**Description**
SM	5	S&P PE ratio	S&P’s 500 Price-to-Earnings
SM	1	VXOCLSx	S&P 100 Volatility Index

Table 12 – Description of the groups cited in Table 11.

Group	**Description**
OI	Output and Income
LM	Labor Market
H	Housing
COI	Consumption, Orders and Inventories
MC	Money and Credit
INTFX	Interest and Exchange Rates
P	Prices
SM	Stock Market

Table 13 – Description of the transformations cited in Table 11 applied to each series x_t. The differencing operator is denoted by Δ.

Transformation	**Description**
1	No transformation
2	Δx_t
3	$\Delta^2 x_t$
4	$\ln x_t$
5	$\Delta \ln x_t$
6	$\Delta^2 \ln x_t$
7	$\Delta(x_t/x_{t-1} - 1)$