Erasmus University Rotterdam

Erasmus School of Economics

Bachelor Thesis BSc^2 Econometrics/Economics

Using Deep Learning in a Big Data Setting for Macroeconomic Forecasting of the Gross Domestic Product of the United States

Author:
Sean Goedgeluk
Student Number:
443106

Supervisor:
Prof. dr. Ph.H.B.F. Franses
Second Assessor:
dr. C. Cavicchia

July 4, 2021

Abstract

The growing number of variables used in macroeconomic forecasting has stimulated the application of different models in diffusion index forecasting. One still undiscovered field of models are those found in deep learning. This research applies the popular long short-term memory (LSTM) neural network, the convolutional neural network (CNN) and the CNN-LSTM combination to macroeconomic forecasting using a large amount of predictors. The ability of these deep learning models to uncover underlying factors is compared with boosting in a simulated environment. Next to this, the models are used to predict the monthly gross domestic product (GDP) of the United States (US). The results show that the LSTM is useful, and better than the boosting algorithm, in situations with clearly defined underlying factors, and not per se the field of macroeconomic forecasting. Oppositely, the CNN and CNN-LSTM model perform worse than the LSTM model in such simulated environments, when compared to the boosting algorithm, but are better than the LSTM in macroeconomic forecasting. That said, the deep learning models fail to significantly outperform boosting in macroeconomic forecasting of the US GDP, but the results indicate that especially CNNs could be further improved to increase performance. CNN-LSTMs show promising results for longer forecasting horizons, but also require further optimization. Finally, the feature maps of CNNs prove useful in determining which variables are most useful in predicting US GDP.

Keywords: macroeconomic forecasting, deep learning, long short-term memory, convolutional neural network, diffusion index forecasting

Contents

1 Introduction

Every day, policy makers try to accurately predict economic indicators for use in fiscal and monetary policy decisions. Recent developments in technology have introduced a vast amount of new macroeconomic data. This vast amount of data, often referred to as big data, has come with a considerable amount of research, as careful selection of the best predictors is now more important than ever. Bai & Ng (2008) note that this is necessary because the information overload that occurs with many predictors often brings down performance. To facilitate this selection, several techniques, as covered by Kim & Swanson (2018), have been introduced and studied over the years. Among these are the heavily researched techniques of dynamic factor models which reduce the dimensionality by aggregating the most important information of the variables in factors (Bai & Ng, 2008, 2009; Kim & Swanson, 2014a,b, 2018; Stock & Watson, 2002a,b). While these have shown very interesting results, new approaches in the field of machine learning, as surveyed by Stock & Watson (2012) and Kim & Swanson (2014a, 2018), have shown increased accuracy in a number of cases. One of these methods, boosting, is especially interesting, as it allows for pre-selection of variables before factor estimation. First adopted by Bai & Ng (2008), both them and Kim & Swanson (2018) have shown good results for this technique.

While boosting, and other machine learning methods, have shown promising results, a new field of research has made its way into macroeconomic prediction: through the application of advanced neural networks, deep learning models can capture intricate patterns in univariate or low-dimensional multivariate time series with high accuracy (Alaminos et al., 2021; Theoharidis, 2021; Cook & Hall, 2017). Originating in other fields, Sezer et al. (2020) mention that these models allow for automatic feature selection among many inputs, as also done by boosting, and a general-purpose learning approach that can be applied to many current-day analytical problems. This makes them very useful in the setting of macroeconomic forecasting with many predictors as they provide an alternative to the boosting algorithm applied by Kim & Swanson (2018) and Bai & Ng (2009). The most commonly used models in forecasting are the long short-term memory (LSTM) network (Cook & Hall, 2017; Koenecke, 2020), the convolutional neural network (CNN) (Theoharidis, 2021; Cook & Hall, 2017; Koenecke, 2020; X. Yang & Liu, 2021) and a CNN-LSTM combination (Alaminos et al., 2021; X. Yang & Liu, 2021; Jin et al., 2020; Livieris et al., 2020). The survey by Sezer et al. (2020) provides further use cases for each of these models in the field of finance. These results on multivariate cases in other fields offer the potential for promising results for macroeconomic forecasting.

While these models offer extensive use on many macroeconomic variables, including economic growth, interest rates, exports and imports, one of the most important and highly researched economic indicators is the growth of the gross domestic product (GDP), a key indicator for economic development. While important at any given time, it is even more important in times of economic crisis and therefore of considerable value for policy makers. Research using the above techniques primarily focuses on forecasting GDP growth, and other variables, for the United States (US), with Stock & Watson (2002a,b, 2012), Kim & Swanson (2014a,b, 2018), Groen & Kapetanios (2016) and Bai & Ng (2008, 2009) applying a broad range of dimension reduction, variable selection and machine learning techniques in this country. The value of this research does not only lie in the United States, but extends beyond this, as many countries experience a spillover effect from the economic growth in the United States (Maćkowiak, 2007; Georgiadis, 2016; Miranda-Agrippino & Rey, 2020). Bai & Ng (2009) and Kim & Swanson (2018) dive deeper in forecasting US macroeconomic variables through the application of boosting, while Alaminos et al. (2021) and Theoharidis (2021) implemented deep learning using multiple inputs to also predict US macroeconomic indicators. In the current literature no research exists regarding the use of deep learning methods in a big data setting with many regressors.

The aim of this paper is therefore to introduce deep learning models, in the form of the popular LSTM, CNN and CNN-LSTM models, in macroeconomic forecasting. I look at the predictive accuracy of these models compared to the boosting algorithm used by Kim & Swanson (2014b, 2018) and Bai & Ng (2009). This results in the following research question:

To what extent can a long short-term memory (LSTM) neural network, a convolutional neural network (CNN) and a CNN-LSTM model outperform boosting in predicting monthly US GDP?

To answer this question, I will first look into the robustness and correctness of the models by comparing them in a simulation study, which will offer insights into the performance of deep learning models against boosting models in a simulated environment. Additionally, it will help determine the ability of these models to uncover underlying factors in data. As such, the following sub-question helps shape the main research of this paper:

Can an LSTM, CNN and CNN-LSTM uncover simulated underlying factors and outperform boosting in prediction?

After this, the models are compared in an empirical setting of forecasting the US GDP, which allows for a relevant integration into the current set of literature. While predictive ability is of vital importance for macroeconomic researchers, the ability to find the most important variables used in prediction also carries considerable value. As such, the final sub-question is as follows:

How can a deep learning model be used to uncover the most important variables in predicting US GDP growth?

This research adds to the scientific basis by introducing new methods in macroeconomic forecasting. On top of this, the comparison with boosting models shows the potential of systematic and non-linear feature selection by the neural network, compared to the feature selection applied by the boosting algorithm. Lastly, it opens up the field into new methods of macroeconomic forecasting. From the perspective of social relevance, it will allow policy makers and economists to use this new tool in forecasting key economic indicators. This will potentially allow for more accurate fiscal and monetary policies to be applied, and how to adjust these policies for different scenarios.

The motivation for this research stems from the fact that deep learning has shown great potential in a range of financial time series, as shown by Sezer et al. (2020), and other fields, but has only marginally been researched in the field of macroeconomic forecasting. While univariate applications (Theoharidis, 2021) and multivariate applications with a small number of predictors (Alaminos et al., 2021; Cook & Hall, 2017) have been researched, these models have not yet been applied in a setting with a large amount of predictors.

The rest of this thesis is structured as follows. Section 2 contains a review on the literature surrounding macroeconomic forecasting, deep learning and US economic growth. After this, I cover the theoretical framework of the models in Section 3. This is followed by the methodology and data in Section 4. Lastly, the results and discussion thereof will be provided in Section 5, while Section 6 concludes and provides opportunities for future research.

2 Literature Review

Macroeconomic forecasting has a long history. With the increase of data, researchers introduced several new models for forecasting. This section will briefly cover the background of macroeconomic forecasting and boosting. Next, the US economic growth is covered, after which deep learning models in a temporal setting are discussed, where the focus lies on the long short-term memory (LSTM) neural network, the convolutional neural network (CNN) and an CNN-LSTM combination. For a more in-depth discussion of the mathematical and theoretical framework of the different models and components used in this research, I would like to refer to Section 3.

2.1 Macroeconomic Forecasting with Many Predictors

With the increase in the availability of macroeconomic variables, more data can be used for macroeconomic forecasting. Stock & Watson (2002b) note that during the early days of this development, the existing econometric models would only incorporate a small number of variables. While the use of variable selection methods can pre-select important variables from the set of predictors, it would still significantly lack the information present in a large data set. To simplify this high dimensional problem and allow more information to be used in forecasting, Stock & Watson (2002b) therefore introduce a method that extracts the covariability through a small number of unobserved latent factors and uses these factors to forecast key macroeconomic variables. The forecast is related to the factors through a linear relationship that also includes dependent variables commonly used in prediction. Stock & Watson (2002b) show that this two-step approach, referred to as index diffusion models, performs well against a large range of benchmarks. In a related article, Stock & Watson (2002a) add to this model by showing consistency and asymptotic efficiency. They furthermore show that the models are consistent during temporal instability, opening this model up to many macroeconomic variables and situations. Lastly, they look at the finite sample performance of these methods through a Monte Carlo simulation study.

Since the introduction of index diffusion models, several researchers have used this setup in combination with new machine learning and factor estimation methods. However, Bai & Ng (2009) discuss several problems with factor estimation and machine learning methods, and introduce the concept of boosting in diffusion index forecasting. Their motivation is twofold. First, they describe that the information criteria which are used in diffusion index models assume that the factor components are ordered based on the explained variance they offer for the entire set of predictors. As a result, choosing the factors that are used in predicting a certain variable are based on which factors are important for the entire set of macroeconomic predictors, and not based on the actual variable that is predicted. However, two different dependent variables might have two different sets of best factors for prediction. In earlier work, Bai & Ng (2008) already showed that different sets are highly likely for two different dependent variables. Second, they note that the nature of diffusion index models does not allow a factor to be used in absence of its preceding factors, as the factors are ordered by importance of explaining the set of predictors. On top of this, functions of the factors are not possible in the current setup, which limits the flexibility in choosing which factors and lags are included in the model. These problems all happen because there is no easy way to select predictors using a small number of regressions without imposing structure on the predictors (Bai & Ng, 2009).

To solve this issue, Bai & Ng (2009) found a model-fitting method that performs subset variable selection on a large set of potential predictors but without relying on *a priori* ordering: boosting; a method previously used in genes and spam data analysis. This method was first introduced by Freund & Schapire (1999) for use in classification problems. The method, as originally proposed, estimates an unknown function such as the conditional mean using a large amount of stage-wise regressions. Shortly after the introduction, J. Friedman et al. (2000) improved the original method and later J. H. Friedman (2001) introduced "L_2Boosting". This method refits base learners to residuals from previous iterations under quadratic loss. Bühlmann & Yu (2003) slightly adjusted this algorithm by fitting the learners with only one predictor at a time from a large set of variables. This last algorithm was modified by Bai & Ng (2009) to handle time-series data in diffusion index prediction.

With the development of new machine learning methods, Kim & Swanson (2014a, 2018) compared the predictability of several combinations of factor estimation and machine learning techniques. Their study is one of the first to not only apply principal component analysis (PCA) but also introduce several other factor estimation techniques in macroeconomic research. These include independent component analysis (ICA), which aims to find independent components over mere uncorrelated components, and sparse principal component analysis (SPCA), which allows for better interpretation of the components by placing (zero) restrictions on various factor loadings. These three factor estimation methods are then combined with machine learning techniques used for variable selection and shrinkage. Their idea is that these methods can (pre-)select the most important macroeconomic predictors and/or principal components and add to the parsimony and accuracy of the models. Their extensive comparison study combines these two parts in various ways, for example by first applying factor estimation followed by the machine learning, or vice versa, and by including lags of the variables, or not.

2.2 Economic Growth in the United States

While macroeconomic forecasting can be applied in a range of cases, properly forecasting economic growth of the United States has always carried significant value. Largely due to the fact that the United States (US) economy continues to be the largest economy in the world measured in gross domestic product (GDP). On top of this, it has long been one of the fastest growing economies, for which GDP growth is one of the most important indicators. With a GDP of 21.433 trillion US dollars[1] (USD) and growth rate of 2.16%, it contributed approximately 20% of total global

[1]Data retrieved from The World Bank at https://data.worldbank.org/

output in 2019 (Economics, 2020). As a traditionally very open and advanced economy, it also continues to be engaged in extensive trade relationships with a large list of countries. Arora & Vamvakidis (2006) delve into the impact of these relationships on economic growth and show that these not only benefitted the US itself, but have also caused extensive economic growth in countries around the world. These countries do not only include developed nations, but also developing and emerging countries. Arora & Vamvakidis (2006) continue by showing that up to 2006, and potentially beyond, the influence of US growth dominates other global shocks that affect growth across different countries. As a result, the predictability of economic growth in the United States is not only important for domestic policy makers, but also for many other countries.

Traditionally, especially before the 2009 crisis, the economic growth was mostly fueled by increasing global integration, and an increasing focus on research & development (R&D) and other technological advances (Economics, 2020). On top of this, Fernald & Jones (2014) note that increased educational attainment and a steep population growth allowed for increased productivity per employee and a higher general output. In combination with a surge in high-tech companies, this created a long-lasting economic boom in the 1990s; the longest economic expansion the US has ever seen. During this time they positioned themselves as an economy dominated by services-oriented sectors, which accounts for 80% of its output (Economics, 2020). It continues to be one of their most important trade areas, in which they rank first in the world based on absolute output. As a result of these developments, the US has become an important trade partner in the world, with net exports of 1.51 trillion USD and net imports of 2.38 trillion USD in 2019, making them second and first in the world, respectively[2]. Their negative trade balance of 870 billion USD makes them an important source of revenue for other countries in the world. Looking at the different industries within the US, the most important export products are found to be Refined Petroleum ($84.9B), Crude Petroleum ($61.9B), Cars ($56.9B), Integrated Circuits ($41.4B), and Vehicle Parts ($41.2B), and they mostly export to Canada ($252B), Mexico ($235B), China ($103B), Japan ($70.1B), and Germany ($59.8B). An overview of the exports in 2020 is provided in Figure 1 below.

The economic importance of the US has made its fiscal and monetary policy a very important economic indicator, and not only for itself. Since the devastating economic recession that ended in 2012, the US has implemented a combination of expansionary fiscal and monetary policy; preventing further economic downturn through tax cuts and stimulus packages (Economics, 2020). While monetary policy has large effects on the US economy, it has also been shown to induce comovements

[2]Data retrieved from the Observatory of Economic Complexity at https://oec.world/en/profile/country/usa.

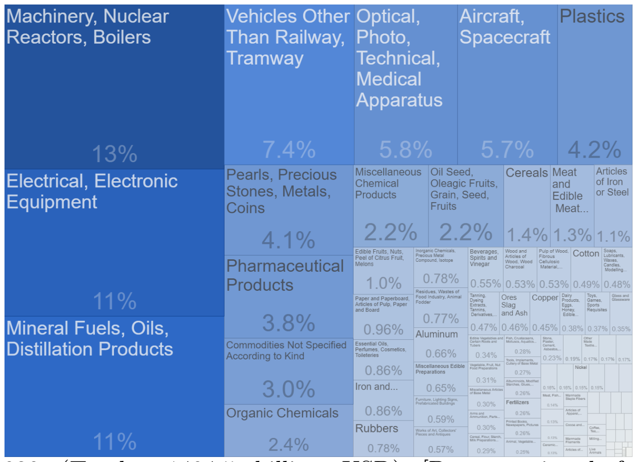


Figure 1: US Exports 2020 (Total: 1404.5 billion USD) [Data retrieved from Trading Economics at https://tradingeconomics.com/united-states/exports-by-category.]

and spillover effects in many other markets. Before the financial crisis, Maćkowiak (2007) shows this effect in emerging markets, in which the US monetary policy quickly and strongly affects interest and exchange rates. Similar effects are shown in later research by Obstfeld (2019). Furthermore, the price level and real output respond by more than these levels in the US itself (Maćkowiak, 2007). More recent work by Georgiadis (2016) additionally shows sizable output spillovers generated by US monetary policy and delves deeper in the underlying causes, noting possible solutions. Miranda-Agrippino & Rey (2020) add to this by defining the 'Global Financial Cycle' which reflects the comovements induced by US monetary policy shocks. They give explicit examples of the effects of such shocks, such as a decline in the provision of domestic credit globally, strong retrenchments of international credit flows, and tightening of foreign financial conditions, which all follow after a monetary contraction in the US (Miranda-Agrippino & Rey, 2020).

The high economic growth seen before the economic crisis has slowed down over the years since and will face several challenges in the upcoming years. Fernald & Jones (2014) primarily attribute this to a decrease in the traditional growth factors. They specifically mention a slower growth in R&D intensity, due to cherry picking of easy developments in earlier years, and slowing population growth. Next to this, the rise of China, India and other emerging economies will continue to shift economic importance away from the US and towards these countries. Changes in humanity's focus on income inequality, climate change and health care might also impact future economic growth (Fernald & Jones, 2014). On top of these internal and external effects, a growth in economic tension has caused several countries and the United Nations (UN) to implement drastic economic sanctions on various countries. An example is the trade war between the US and China during the Trump presidency. Neuenkirch & Neumeier (2015) show that economic sanctions by the US cause severe decline in economic growth both domestically and abroad. The opposite is also highly feasible, especially by new world powers such as India and China. This adds to the increasing global effect that economic changes within countries have on economic growth in- and outside the country in question, especially those of such stature as the US. As said before, this makes economic prediction of great importance for a large amount of countries.

2.3 Forecasting using Deep Learning

2.3.1 Long Short-Term Memory (LSTM) Neural Network

The LSTM Neural Network, as first introduced by Hochreiter & Schmidhuber (1997), is a type of recurrent neural network (RNN) designed for temporal forecasting. This commonly used network architecture has been applied for forecasting in a broad range of fields, including many areas of finance (Sezer et al., 2020; Selvin et al., 2017; Koenecke, 2020; Jin et al., 2020), energy (Bedi & Toshniwal, 2019; Qing & Niu, 2018) and traffic (Zhao et al., 2017). This includes both univariate, as well as multivariate cases. Within macroeconomic forecasting, this approach has been used in a low-dimension multivariate setting for unemployment by Cook & Hall (2017). Their approach has shown good results for the LSTM, which offers potential for settings with many predictors. This is supported by the results in the wide range of applications already mentioned.

2.3.2 Convolutional Neural Networks

Another commonly used type of neural network is the convolutional neural network (CNN). Traditionally, this architecture was primarily used for image and video recognition/classification tasks and natural language processing, but it has since expanded into other applications. Among these applications are time series classification, for example the research done by Zhao et al. (2017), and time series forecasting, both in a range of scientific fields. The robustness and adaptability of these models in real world time series is shown by Zhao et al. (2017) and Zhou (2018), which is further supported by X. Yang & Liu (2021) and a broad range of literature that uses CNNs for time series prediction. Examples are, among others: univariate revenues (Koenecke, 2020), univariate stocks (Selvin et al., 2017), multivariate data to predict the S&P500 (Borovykh et al., 2017) and multivariate human activity recognition using body-worn sensors (J.-B. Yang et al., 2015). Additionally, Sezer et al. (2020) provide a broad overview of the use of CNNs, and other deep learning models, in the most important topics in finance.

While CNNs have seen more and more practical use-cases and interesting developments, the application to macroeconomic forecasting has only recently seen its introduction. One notable contribution is the research by Cook & Hall (2017), who apply a CNN, and several other neural network architectures, to forecast the US unemployment rate using the time series itself, as well as differenced series. This is extended by Alaminos et al. (2021), who use a recurrent adoption of the standard CNN to predict GDP growth rates for a large number of countries. To achieve this, they use 24 macroeconomic indicators. While the results in the case by Cook & Hall (2017) show poor results for the CNN, as it is outperformed by an LSTM, the multivariate case by Alaminos et al. (2021) shows promising results for the CNN, as it is significantly better than four other deep learning models and not outperformed by any of the other two. These results show potential for cases where there are many more possible regressors.

With the increasing use of CNNs in time series forecasting, both univariate and multivariate architectures have been developed, which are both discussed by X. Yang & Liu (2021). While they focus on non-stationary time series, these models generalize to stationary data. For their CNN architecture they focus on the Lenet-5 architecture, which is introduced by Lecun et al. (1998) and has shown promising results in predicting the Dow Jones Industrial Average (DJIA) (X. Yang & Liu, 2021). Similar models are used by Liu et al. (2020) and Cao & Wang (2019), who both show results that support the use of CNNs. While not in the field of macroeconomic forecasting, the research done by J.-B. Yang et al. (2015) offers additional insight in the use of CNNs and multivariate time series input. They use multiple signals from body-worn sensors as temporal input, which can instead be a series of macroeconomic indicators.

2.3.3 CNN-LSTM Architecture

The temporal characteristics of an LSTM can be combined with the convolution characteristics of the CNN. By feeding the output of a CNN into an LSTM, the horizontal relationship between the time series as extracted by the CNN can persist over time in the LSTM (Jin et al., 2020). This model is further described by X. Yang & Liu (2021), who implement it to predict the DJIA. The approach is also commonly used in other time series as shown by Jin et al. (2020), who experiment with it on Beijing meteorological data, and by Livieris et al. (2020) to predict gold prices. In macroeconomic forecasting the available literature is quite limited, with Alaminos et al. (2021) following a similar architecture but instead using a regular RNN instead of an LSTM, and Theoharidis (2021) using it to predict inflation. Due to their good results, this architecture shows potential in macroeconomic forecasting using a high number of predictors.

3 Theoretical Framework

3.1 Diffusion Index Models

Stock & Watson (2002b,a) introduced the notion of diffusion index models in macroeconomic forecasting. In their approach, they first extract a set of r common latent factors F_t based on the notion that X_t can be decomposed in F_t by the equation:

$$X_t = \Lambda F_t + e_t, \tag{1}$$

using a certain factor decomposition method. Here, t is the time with $t = 1, ..., T$ and j corresponds to the index of the predictor with $j = 1, ..., N$. Then, X_t is an N-dimensional set of candidate predictors for a time series y_t and e_t is an $N \times 1$ vector of idiosyncratic disturbances. The combination of these factors and several observed macroeconomic predictors, such as the lags of the independent variable, are then used to forecast future values of y_t through the linear relationship:

$$y_{t+h} = \beta_F' F_t + \beta_w' w_t + \varepsilon_{t+h}. \tag{2}$$

Here, h is the forecast horizon, w_t is an $m \times 1$ vector of observed variables and ε_{t+h} is the resulting forecast error. Because e_t is unlikely to be cross-sectionally independent and temporarily independent in this macroeconomic forecasting application, Stock & Watson (2002b,a) allow for these error terms to be serially correlated and (weakly) cross-sectionally correlated. Stock & Watson (2002a) delve into the theoretical implications and solutions of this assumption and find that general estimation and decomposition methods, such as principal component analysis (PCA) explained in Section 3.2, can be applied without loss of consistency or other important characteristics.

3.2 Factor Estimation and Principal Component Analysis

This section will provide a brief overview of factor estimation and principal component analysis (PCA). Additional details on these techniques are covered by Bai & Ng (2008, 2009), Kim & Swanson (2014a) and Stock & Watson (2002a, 2012). Factor models are used to extract a set of common factors r ($r \ll N$) from a high-dimensional set of N predictors, following the idea that this set of predictors can be represented as in Equation 1, to lower the dimension of the data. Here, the product of the factor loadings Λ_j (the jth row of Λ) and common factors F_t is called the common component of X_{tj} (the jth variable of X_t), which corresponds to the factor representation of the data.

A well-known technique to find these common factors F_t is the use of principal components. PCA uses eigenvalue decomposition to find orthogonal, thus uncorrelated, factors. Before any steps of the algorithm are applied, it is very common to normalize the data by subtracting the mean and dividing by the standard deviation. Following this, the first step in PCA is to estimate the covariance matrix $\hat{V}$ (or correlation matrix) of X_t. This matrix is used in the second step to calculate the eigenvectors e_j and corresponding eigenvalues λ_j, following the equation $\hat{V}e_j = \lambda_j e_j$, using singular value decomposition (SVD). Given the eigenvalues and -vectors, the principal components are given by:

$$P_{tj} = e_j' X_t, \tag{3}$$

where $j = 1, ..., N$. The factors determined through PCA can be ranked based on the amount of variance they explain in the dataset, as PCA tries to explain as much of the variance of the remaining data in each subsequent principal component. This allows one to select the amount of principal components to use based on a pre-defined level of explained variance or another selection criterion. The latter is commonly used in diffusion index forecasting, as also used by Bai & Ng (2002, 2008, 2009) and Kim & Swanson (2014a, 2018). They use the criterion function $SIC_3(r)$ suggested by Bai & Ng (2002):

$$SIC(r) = V\left(r, \hat{F}\right) + rh(N,T) = V(r,\hat{F}) + r\hat{\sigma}^2\left(\frac{(N+T-r)\ln(NT)}{NT}\right), \tag{4}$$

where $h(\cdot)$ is a penalty function, provided in Equation 4, and $V(\cdot)$ is a function minimizing the distance between the original variables X and their factor representation. The latter follows the implementation by Bai & Ng (2002) and described as:

$$V\left(r, \hat{F}\right) = \min_{\Lambda} \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(X_{it} - \lambda_i^{r\prime} \hat{F}_t^r\right)^2. \tag{5}$$

Kim & Swanson (2018) add an additional limit of r_{max}=20 to the number of factors used in their empirical study, which this research copies.

3.3 Boosting

While originally used in classification problems and later in regression problems, Bai & Ng (2009) use this method to pre-select the most important variables among a high-dimensional list of predictors for use in factor augmented autoregressions. Conceptually, boosting builds on a set of many weak learners and uses this set repeatedly on changed data. A linear combination of these weak learners is the output of this technique. Similar to Kim & Swanson (2014a,b, 2018), I implement the "Component-Wise L_2Boosting" algorithm by Bai & Ng (2009) as presented in Algorithm 1 below. This is the algorithm as described in Kim & Swanson (2014b), with the addition of updating the estimator $\hat{\beta}_i$, as described by Bai & Ng (2009), in the last step of the algorithm. Before applying the boosting algorithm, Bai & Ng (2009) and Kim & Swanson (2014b, 2018) let $Z = Y - \hat{Y}^W$ obtained from fitting an autoregressive model to the target variable using W_t as regressors.

Algorithm 1 Components-Wise L_2Boosting

```
procedure BOOSTING($Z$,$\nu$,$r$,$M$)
    Initialize $\hat{\mu}_0(F_t) = \bar{Z}$ for each $t = 1, ..., T$ and $\hat{\beta}_0 = \mathbf{0}_r$
    for $i = 1 \rightarrow M$ do
        for $t = 1 \rightarrow T$ do
            Compute $u_t = Z_t - \hat{\mu}_{i-1}(D_t)$
        end for
        for $j = 1 \rightarrow r$ do
            Regress $u$ ($T \times 1$ residuals) on $\hat{F}_j$ ($j$th factor) to obtain $\hat{\beta}_j$
            Compute $\hat{d}_j = u - \hat{F}_j \hat{\beta}_j$
            Compute $SSR_j = \hat{d}_j' \hat{d}_j$
        end for
        Determine $j_i^* = \text{argmin}_{j \in [1,...,r]} SSR_j$
        Compute $\hat{g}_*^i(F) = \hat{F}_{j_i^*} \hat{\beta}_{j_i^*}$
        for $t = 1 \rightarrow T$ do
            Compute $\hat{\mu}_i = \hat{\mu}_{i-1} + \nu \hat{g}_*^i$
        end for
        Compute $\hat{\beta}_i = \hat{\beta}_{i-1} + \nu \hat{\beta}_{j_i^*}$
    end for
end procedure
```

As Bai & Ng (2009) use $0 < \nu < 1$, boosting does not only do variable selection but also shrinkage. Kim & Swanson (2014b, 2018) choose $\nu = 0.5$ and $M = 50$, which is also used in this paper.

While this algorithm has shown great potential, Kim & Swanson (2014b) and Bai & Ng (2009) note the problem of overfitting that may arise if this algorithm is repeated too often. As such, choosing the stopping parameter M correctly in Algorithm 1 is of great importance. Bai & Ng (2009) use the following information criterion to determine M:

$$IC(i) = \log\left[\hat{\sigma}^{t^2}\right] + \frac{\log(T) \cdot df_i}{T}, \tag{6}$$

where $\hat{\sigma}^{t^2} = \sum_{t=1}^{T}\left(Y_t - \hat{\mu}^i(\hat{F}_t)\right)$. Then, using this criterion, M is given by:

$$M = \arg\min_i IC(i) \tag{7}$$

In this setup, the degrees of freedom df_i is defined as $df_i = trace(B_i)$, where $B_i = B_{i-1}\nu\mathbf{P}^{(i)}(I_T - B_{i-1})$ with $\mathbf{P}^(i) = \hat{F}_{j_*^i}\left(\hat{F}'_{j_*^i}\hat{F}_{j_*^i}\right)^{-1}\hat{F}'_{j_*^i}$. Starting values for B_i are given as $B_0 = \frac{1}{\nu}P^{(0)} = \mathbf{1}_T'\mathbf{1}_T/T$, where $\mathbf{1}_T$ is a $T \times 1$ vector of 1s. Using Equation 2, the prediction is then:

$$\hat{Y}_{t+h}^{Boosting} = W_t\hat{\beta}_w + \hat{\mu}^M\left(\hat{F}_t\right). \tag{8}$$

3.4 Deep Learning Models

The core focus of this research lies in the application of deep learning models in a big data setting of macroeconomic forecasting. These models are primarily based on the use of neural networks (NN) in learning complex relationships between input and output data. Based on the literature review in Section 2, three different types of these neural networks for prediction are used in this research. First, I will cover long short-term memory (LSTM) neural networks, followed by convolutional neural networks (CNNs) and lastly a CNN-LSTM combination.

3.4.1 Long Short-Term Memory Neural Network

Neural networks allow complex non-linear relationships to be estimated through the combination of many simple mathematical expressions in a network-like structure. Following the notation by Franses & Dijk (2000), such neural network relationships in time-series can be expressed as:

$$y_t = F(x_t;\theta) + \varepsilon_t, \tag{9}$$

where traditional neural networks model $F(x_t;\theta)$ as:

$$F(x_t;\theta) = \phi_0 + \sum_{j=1}^{q}\beta_j G(x_t'\gamma_j). \tag{10}$$

Here, q corresponds to the number of logistic components, or neurons, used in the network. x_t corresponds to lagged values of y_t, manipulations thereof or other variables that may drive y_t, and $x_t'\gamma_t$ corresponds to a linear combination of the input variables based on the estimated weights γ_j. $G(\cdot)$ is the logistic function given by:

$$G(z) = \frac{1}{1+\exp(-z)}. \tag{11}$$

While these neural networks can estimate a large set of complex relationships in time-series, they are used similarly to neural networks in classification and model each time step $t = 1, ..., T$ based on the estimated coefficients of $F(\cdot)$. That is, they do not take into account any updated information available from previous steps in the time-series process after the model has been estimated. Even when including many lags, it is difficult for these models to capture short- and long-run time series dependencies. To solve this problem, recurrent neural networks (RNNs) were developed. These neural networks have been shown to capture time-dependencies much better by adding a recurrent component to Equation 9, which uses the output of the previous observation. While this already allows short term dependencies to be modelled very well, a slight modification of this model, the long short term memory (LSTM), allows for much longer time-series dependencies to be modelled. This is possible due to the introduction of a saved state, which resembles a complex moving average type of component. The weights and value of this moving average depend on many inputs, but allow complex information about the development of the time series to be stored over a longer period of time. These additions cause x_t to be replaced in $F(\cdot)$, see Equation 10, by the current state of the recurrent component h_t, such that:

$$y_t = \phi_0 + \sum_{j=1}^{q} \beta_j G(h'_{j,t}\gamma_j) + \varepsilon_t, \tag{12}$$

with q recurrent components. The input x_t is instead used in estimating the recurrent value and the saved state, which determine the time-dependence of the model. Taking inspiration from the description by Olah (2015), a single recurrent component h_t can be calculated by using a linear combination of the current input x_t and the output of the recurrent component one time-step ago h_{t-1}, as well as the saved state C_t. This results in:

$$h_t = G(\phi_{h0} + \beta'_{h1}x_t + \beta'_{h2}h_{t-1}) \cdot \tanh(C_t), \tag{13}$$

where $\tanh(\cdot)$ corresponds to the hyperbolic tangent function given by:

$$\tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^{2z}-1}{e^{2z}+1}, \tag{14}$$

and β'_{h1} and β'_{h2} to the weight parameters of the linear combinations $\beta'_{h1}x_t$ and $\beta'_{h2}h_{t-1}$ respectively. The $\tanh(\cdot)$ function is used here because its range (from -1 until 1) allows the recurrent component to be impacted negatively as might be required within the structure of the time series. Additionally, it has the added benefit of faster conversion with non-linear optimization methods. Finally, the saved state C_t is given by the complex moving average type of structure:

$$C_t = G(\phi_{c00} + \beta'_{c01}x_t + \beta'_{c02}h_t) \cdot C_t + G(\phi_{c10} + \beta'_{c11}x_t + \beta'_{c12}h_t) \cdot \tilde{C}_t, \tag{15}$$

where the first logistic function in the sum determines which part of the old information is forgotten, also called the 'forget gate' in other literature. The second logistic function determines the degree to which the new candidate state $\tilde{C}_t$ is added. This candidate state is given by $\tilde{C}_t = \tanh(\phi_{c20} + \beta'_{c21}x_t + \beta'_{c22}h_{t-1})$, which is also referred to as the 'input gate'. Here, the $\tanh(x)$ function is used to again allow for the possibility of a negative impact of the state update.

Before an LSTM is estimated, the time series should be deseasonalised, as forecasts from neural networks using deseasonalised data are significantly more accurate (Nelson et al., 1999). The preprocessing steps taken by Kim & Swanson (2018) take care of this.

3.4.2 Convolutional Neural Networks

Traditionally used for image and natural language processing tasks, Lecun et al. (1998) introduced the CNN architecture "LeNet-5", which has since been commonly used in many areas, including financial and macroeconomic time series forecasting (see Section 2.3.2). These advanced neural networks allow for images and other 1D/2D signals to be filtered and processed in an efficient way, much more efficient than by a normal feed-forward neural network. To achieve this computational feat, CNNs combine three mathematical ideas: local receptive fields, shared weights (also known as weight replication) and temporal sub-sampling (Lecun et al., 1998).

The "LeNet-5" model consists of several (repeated) components, which are covered in-depth by Albawi et al. (2017). In this paper, the notation is adjusted to match its application in time series forecasting. The most important, and perhaps intricate component of the CNN are convolutions, which use the first two ideas mentioned earlier. A convolutional layer applies several matrix filters $h_q(\cdot)$ to the matrix X_t of regressors, where $q = 1, ..., Q$ refer to the different filters. In this setting, X_t is a $T_s \times N$ matrix that contains a chosen T_s number of lagged values of the N variables at time t. The filters are then applied by subdividing X_t in small sub-matrices of a predetermined size and applying the filters to each sub-matrix separately. This concept is also known as local receptive fields. By using sub-matrices instead of the entire matrix X_t, the number of coefficients in the model is drastically reduced. However, if the coefficients of these filters would be estimated separately for each sub-matrix, the model would still contain many coefficients and estimation would take a considerable time. To further simplify the model, the filters $h_q(\cdot)$, $q = 1, ..., Q$, use the same coefficients for each sub-matrix of X_t, also referred to as shared weights. This decreases the number of coefficients to be estimated even more. A numerical example comparing the number of coefficients between a feed-forward neural layer and a convolutional layer is provided by Albawi et al. (2017). Although it seems that complexity is lost due to these simplifications, the use of different filters allows different features to be extracted from the data (Albawi et al., 2017). The convolutional layer then creates a new matrix of filtered data for each filter $h_q(\cdot)$, $q = 1, ..., Q$, which are often referred to as feature maps. These matrices are then described by:

$$H_{q,m,n} = (X_t \cdot h_q)_{m,n} = \sum_j \sum_k h_{q,j,k} X_{t,m-j,n-k}, \qquad q = 1, ..., Q, \tag{16}$$

with m and n respectively the rows and columns of X_t, and j and k the iterators used to go over each sub-matrix. By combining this for the different filters in a convolutional layer, the complete convolutional layer for the lth layer in the network is described by:

$$\mathbf{Z}_{tl} = \mathbf{W}_l \cdot \mathbf{A}_{t,l-1} + \mathbf{b}_l \quad \text{and} \quad \mathbf{A}_{tl} = g_l\left(\mathbf{Z}_{tl}\right).$$

Here, the three-dimensional matrix $\mathbf{Z}_{tl}$ is an intermediary step that multiplies the input $\mathbf{A}_{t,l-1}$ of the previous layer with the coefficient matrix $\mathbf{W}_l$ and adds a bias term $\mathbf{b}_l$, as commonly found in neural networks. This intermediary step $\mathbf{Z}_{tl}$ is then used as input for the non-linear function $g_l(\cdot)$ to be passed to the next layer. In CNNs, $g_l(\cdot)$ most commonly corresponds to the ReLU function (Albawi et al., 2017), given by:

$$g_l(x) = x^+ = \max(0, x). \tag{17}$$

The second important component used in CNNs, introduced by Lecun et al. (1998) and described in detail by Albawi et al. (2017), is the use of pooling to down-sample the data between the different convolutional layers. This happens without affecting the number of filters, which contain important information about the features of the data. While there are different types of pooling, the most common is max-pooling, This method partitions the output matrix Z_{tl} of a previous layer in sub-regions of a predefined size and returns the maximum value of the sub-region.

The LeNet-5 architecture uses a convolutional layer followed by a max-pooling layer. This combination of the two layers is repeated once, after which the data from the last layer is flattened and fed into one or more fully connected layers.

4 Data and Methodology

This research uses both a simulation and an empirical study to compare the models. First, this section covers the models that will be implemented, after which the data is discussed.

4.1 Simulation

To investigate the ability of deep learning models to uncover underlying factors in data, I implement a simulation study similar to the simulation approach 'GDP 1' (data generating process 1) implemented by Bai & Ng (2009), which they use to test their boosting algorithm. Following their approach, I first generate k underlying factors using an AR(1) model, after which I use a factor-based construction of the observed X_t ($T \times N$) variables using the generated factors. This corresponds to the following data generating process (DGP) for $t = 1, ..., T$:

$$X_{it} = F_t \lambda_i' + \sqrt{k}\varepsilon_{it}, \qquad i = 1, ..., N, \quad \text{and} \tag{18}$$

$$F_{jt} = \alpha_j F_{j,t-1} + u_{jt}, \qquad j = 1, ..., k. \tag{19}$$

Here, I use $(\varepsilon_{it}, u_{jt}) \sim N(0, I_2)$, following Bai & Ng (2009), and $\alpha = (0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3)'$. For the factor loadings λ_i (ith row of λ), I follow Bai & Ng (2009) and use $\lambda_i = 0.5 \cdot N(0, k)$. The predictive accuracy is evaluated by creating a target series y, for $t = 1, ..., T$ from the generated factors as follows:

$$y_{t+h} = F_t \beta_F + e_{t+h}, \tag{20}$$

where $e_{t+h} \sim N(0, 1)$ and the factor loadings $\beta_F = (0.8, 0.5, 0.3, -0.1, -0.3, -0.6, -0.9)'$.

To provide an extensive comparison of the deep learning models with the boosting algorithm, I generate 100 data sets for the combinations of the following parameters: k =3, 5 and 7; T =50, 200 and 500; and N =10, 50 and 100. I start at $T = 50$ and $N = 10$ because the CNN model requires a large enough input matrix X_t of time series data. If X_t is smaller than the kernel size, then the CNN will not be able to train on the data.

4.2 Empirical Data

As commonly used in the diffusion index literature, I use monthly observations from 144 US macroeconomic time series for the same time frame as Kim & Swanson (2014b, 2018). The data ranges from January 1960 until May 2009 and is retrieved from Wharton Research Data Services (WRDS) and verified with the data used by Kim & Swanson (2018)[3]). From these macroeconomic time series, I focus on predicting the growth rate of the gross domestic product (GDP), for which I take the log-difference of the GDP. This approach is the same as used by Kim & Swanson (2014b, 2018). A plot and summary statistics of this data are presented in Figure 2 and Table 1 respectively. The plot also contains a three-year (red) and ten-year rolling average, which shows that the growth rate has gone down only very slightly in the entire period, yet the volatility around this mean seems to have decreased.

Table 1: Summary statistics of the US GDP growth rate from January 1960 until May 2009

T	N	Mean	Median	Std. Dev.	Min	Max	Skewness	Kurtosis
593	144	0.256	0.265	0.440	-1.354	2.031	-0.217	0.964

Note. Growth rates presented in percentages.

As most of the other macroeconomic variables are also non-stationary, preprocessing is applied across all time series. As different structures are seen in the time series, different techniques are applied depending on the time series at hand. A description of the variables, including the preprocessing approach, is provided in Section 6 in Appendix A. This data is then used in two settings. First, the same initial time frame and expanding window estimation as the approach by Kim &

[3]Kim & Swanson (2018) and other researchers use this data for a larger range of academic literature, and can be found at https://sites.google.com/site/khdouble2/research (Kim & Swanson, 2018)

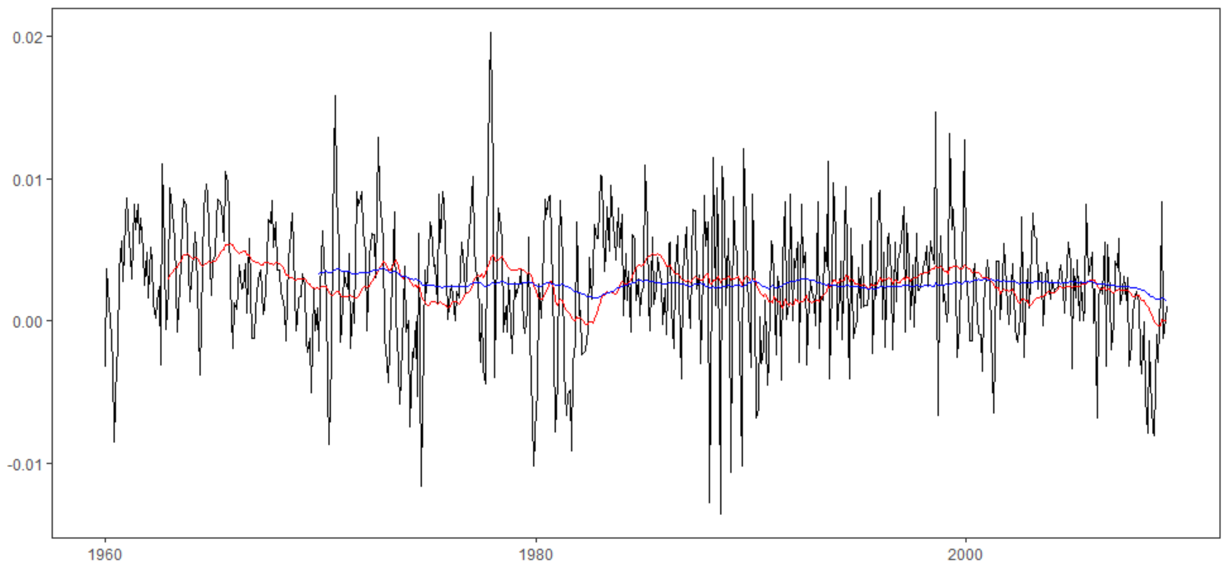


Figure 2: US growth rates from January 1960 until May 2009, including a 3-year (red) and 10-year (blue) moving average

Swanson (2018) are used, to verify correct implementation of the boosting method. This corresponds to an expanding window starting with predicting after December 1972. As the main goal of this research is the application of deep learning models in the context of macroeconomic forecasting, the second settings follows a deep learning-based concept of training and test data for the second approach. This is because neural networks generally need a much longer training period to perform well. As the different deep learning models use different windows as their input, the focus lies on creating a test set with an equal length for all models. This allows for the different methods to be compared where necessary. To facilitate this, I predict the last 30% of the data for all models and horizons, which corresponds to predicting the values from August 1990 until May 2009. As a result, the training period consists of less than, but approximately, the first 70% of the data from January 1960 until July 1990. Due to the extensive time deep learning models require for training and the focus on hyperparameter optimization and model comparison, the deep learning models will be trained only once on the training period after which predictions are made for the entire test period. The boosting and autoregressive model, see Section 4.3 below, will be re-estimated for each time-step as is customary in diffusion index forecasting and also similar to Kim & Swanson (2014a,b, 2018).

4.3 Models

For this research, I will implement five models. Two of these models will be used as benchmarks and the other three will be three different deep learning models. This section will first discuss the benchmark models, followed by an in-depth discussion regarding the choice of deep learning models.

4.3.1 Benchmark Models

The first benchmark model is an autoregressive (AR) model, which is the main benchmark model used by Kim & Swanson (2014a,b, 2018). I follow their idea by implementing a univariate AR(p) computed as $\hat{Y}^{AR}_{t+h} = \hat{\alpha} + \hat{\phi}(L)Y_t$, where p is the number of lags selected with the Bayesian information criterion (BIC). It is important to note that two implementations of an AR model are possible: either by only applying ordinary least squares (OLS) regression, as implemented by Kim & Swanson (2014a,b, 2018), or by a combination of OLS and maximum likelihood (ML) estimation. After a brief review of the performance, the latter implementation was found to be better. As such, the forecasts of this method are used when comparing it to the deep learning models. While the primary goal is to compare deep learning to boosting, it is interesting to also compare it to the performance of a common autoregressive model. The AR model is implemented using the programming language Python and the *statsmodels* library by Seabold & Perktold (2010). On top of this, the *scikit-learn* library by Pedregosa et al. (2011) is used, which is also important for many of the other models and processing steps in this paper.

For the second model, I opt for the L_2 boosting algorithm covered in Section 3.3 and as used by Kim & Swanson (2014a,b, 2018), for which I will use the SP2 approach by Kim & Swanson (2014a,b, 2018). The SP2 approach first uses the Boosting algorithm, see Section 3.3, to construct a subset of the predictors after which factors are constructed using PCA. Diffusion index forecasting is then used to make predictions, as outlined in Section 3.1. This model is implemented following the methodology outlined by Kim & Swanson (2014a,b, 2018) using Python.

4.3.2 Deep Learning Models

The three deep learning models that I will implement are an LSTM architecture, a CNN architecture and a CNN-LSTM combination. More details about the underlying architecture can be found in Section 3.4.

The LSTM will be designed as follows: one observation for all 144 variables will each be fed into a recurrent component in an LSTM layer, using the activation functions given in Section 3.4. Of these components, a random percentage of the recurrent components are turned off using a dropout layer to prevent overfitting, as is common with these type of neural networks. The number of neurons in the LSTM and the dropout rate are determined through hyperparameter tuning discussed in Section 4.3.3 below. The LSTM layer then feeds into an output node corresponding to the predicted value.

The second model, the convolutional neural network (CNN), follows the LeNet-5 architecture described in Section 3.4. Following this architecture, I use two serially connected combinations of a convolutional layer and a max-pooling layer, after which two fully connected layers and a linear output layer are added. The convolutional layers both have a filter size of 3×3 and the number of filters is varied using hyperparameter tuning. Meanwhile, the max-pooling layer will use a pooling size of 2×2. The input to this model will be three years of past data (36 months) for all 144 macroeconomic variables, resulting in an input matrix X_t of size 144×36. For the simulation study the input will be 10, 20 and 50, for respectively T equal to 50, 200 and 500. The fully connected layer has sixteen neurons.

Lastly, I implement a CNN-LSTM combination, following similar approaches by X. Yang & Liu (2021), Livieris et al. (2020) and Jin et al. (2020). The architecture of the CNN part follows the same model as the regular CNN above, after which the output of the last hidden max-pooling layer feeds into an LSTM instead of a fully connected layer. The LSTM then feeds into a fully connected layer and an output node. The number of filters of the CNN are determined using hyperparameter tuning, while the number of neurons of the LSTM is fixed at 32 and that of the fully-connected layer at 16.

All these models are implemented using Python and the *TensorFlow* library by Abadi et al. (2015), a common library for the development of (advanced) neural networks. While the implementation for the CNN and LSTM is fairly straightforward in this library, the CNN-LSTM architecture requires the use of a *TimeDistributed* layer for the CNN and max-pooling layers. This adds a time component to this part of the architecture, which is required to make the connection to the LSTM.

4.3.3 Hyperparameter Tuning

As the performance of deep learning models is highly dependent on the hyperparameters of the architecture, such as the number of neurons (LSTM) and filters (CNN), hyperparameter tuning is key to developing a sound model. The most basic method for hyperparameter optimization is grid search, which refers to testing all hyperparameters options provided by the user to find the best set of parameters. For cases where speed and efficiency is important, other methods have been developed that skip several parameters by assuming a distribution of the performance of the neural network. As efficiency is key in the simulation study of this paper, I use the best model based on the first data set to forecast the data for all data sets in the particular series; hyperparameter tuning for each data set would be too computationally heavy and goes beyond the purpose of this research. To speed up hyperparameter optimization, I opt for the Hyperband search algorithm introduced by Li et al. (2018). This algorithm speeds up random search through adaptive resource allocation such as iterations, samples or features to randomly sampled hyperparameters. Li et al. (2018) show that Hyperband provides over an order-of-magnitude speedup compared to Bayesian optimization, which is very desirable for simulations. As I want to delve deeper into the effects of the hyperparameters in the empirical case, I apply a grid search in this part of my research. The hyperparameter options used in both the empirical and simulation study can be found in Table 2 below.

Table 2: Hyperparameters used in grid search (empirical study) or Hyperband optimization (simulation study) for the deep learning models

Panel A: Empirical Study (Grid Search)											
LSTM						CNN and CNN-LSTM					
Num. LSTM Neurons	16	32	64	128	256	Num. Filters Layer 1	4	8	16	32	64
Dropout Rate	0.1	0.2	0.3			Num. Filters Layer 2	4	8	16	32	64
Epochs	100					Epochs	50				
Panel B: Simulation Study (Hyperband Optimization)											
LSTM						CNN and CNN-LSTM					
Num. LSTM Neurons	[16,32,...,240,256]					Num. Filters Layer 1	4	8	16	32	64
Dropout Rate	[0.1,0.15,0.2,0.25,0.3]					Num. Filters Layer 2	4	8	16	32	64
Epochs	50					Epochs	25				

For the empirical study, I opt to vary the number of filters for the CNN layers between 4 and 64 for both layers of the CNN and CNN-LSTM. For the latter model, I choose to keep the number of recurrent components in the LSTM constant at 32. The LSTM model will use hyperparameter tuning for the number of recurrent components in the LSTM layer, ranging from 16 to 256, and the dropout rate of this layer. In the simulation study, I optimize the same parameters in the same range, but allow for more more options in-between the range for the LSTM, as seen in Panel B of Table 2. The batch size for the empirical study is set at 64, which is reasonable for the size of the data set, while the batch size for the simulation study is set at $T/10$, as the batch size should not be too large compared to the number of observations in a data set.

All these hyperparameters are of great influence to the performance of the neural networks. While other options are possible, such as the learning rate of the optimizer and the number of epochs, I leave these for potential future research. As it is known that an increase in epochs usually increases performance, with some exceptions, I choose the number of epochs at a considerably large value such that convergence can be achieved, see Table 2, and do not use hyperparameter tuning for the number of epochs. For the simulation study, I only perform optimization for the first data set and use the parameters from the best model for all other data sets. Hyperband optimization is contained in *TensorFlow* by Abadi et al. (2015).

4.4 Forecast Comparison

The models will be used to forecast for three different horizons following Kim & Swanson (2018): one, three and twelve time steps ahead. I will compare these forecasts using the mean square forecast errors (MSFE), as applied by Kim & Swanson (2014a,b, 2018). The MSFE implemented by Kim & Swanson (2014a,b, 2018) is given by:

$$\text{MSFE}_{i,h} = \sum_{t=R-h+2}^{T-h+1} \left(Y_{t+h} - \hat{Y}_{i,t+h} \right), \tag{21}$$

where $\hat{Y}_{i,t+h}$ is the forecast at t of model i for forecast horizon h, while Y_{t+h} is the actual observation at $t+h$, the forecasted value. R is the last index for which no value is predicted.

The MSFE will be used to statistically test the accuracy of the deep learning models against the AR and boosting model by using the Diebold-Mariano (DM) test as introduced by Diebold & Mariano (1995) and also used by Kim & Swanson (2018). The test statistic is defined as DM $= \bar{d}/\sqrt{(\text{Var}(\hat{d}_{t+1})/\text{T})}$, where $\bar{d}$ is the sample mean of $d_{th} = L(\hat{y}_{thi}, y_t) - L(\hat{y}_{thj}, y_t)$. This is the difference of a loss function computed with estimates of $\hat{y}_{thi}$ and y_t for comparison of model i and j. Here, h refers to the h-step-ahead forecast estimated for the observation at time t or, differently stated, $\hat{y}_{thi}$ is the forecast h steps ago. Under H_0 of equal accuracy and several weak conditions, DM $\xrightarrow{d} \mathcal{N}(0,1)$ and $\text{Var}(\hat{d}_{t+1})$ can be calculated from the sample variance, as described by Diebold & Mariano (1995). This allows for a direct statistical comparison of the accuracy of two models using the MSFE as in Equation 21. I use the DM-test with a two-sided alternative hypothesis H_1 of a different accuracy for, separately, the LSTM, CNN and CNN-LSTM model compared to the AR and boosting model. For the simulation study, I count the number of times the DM-test is rejected in favor of the deep learning for the different data sets. The test is performed at an α-level of 10%.

5 Results and Discussion

5.1 Simulation

The results of the deep learning model comparison to boosting are presented in Table 3 below. As the main research goal is to compare the deep learning models to boosting and because boosting was found to be worse than the AR model, the comparison with AR is omitted but can be found in Table 7 in Appendix B. The best parameters for each scenario, as found by using Hyperband optimization, can be found in Table 6, also in Appendix B.

Table 3: Average MSFE ratios and number of DM-tests that showed significantly better performance for the deep learning models compared to the boosting algorithm for the simulation scenarios. *(E.g. 0.598 in the 2th column, and 4th row corresponds to a 40.2% better accuracy for the LSTM model compared to the boosting model, for the simulation scenario with $K = 3$ underlying factors, $N = 10$ variables and $T = 50$ number of observations.)*

Panel A: LSTM

	K=3			K=5			K=7		
Num. Obs.	N=10	N=50	N=100	N=10	N=50	N=100	N=10	N=50	N=100
50	0.598 (54)	0.468 (53)	0.464 (59)	1.136 (22)	0.809 (29)	0.631 (53)	1.531 (13)	0.788 (24)	0.808 (3)
200	0.522 (93)	0.444 (100)	0.418 (98)	0.832 (39)	0.603 (85)	0.591 (86)	0.785 (57)	0.591 (95)	0.576 (99)
500	0.469 (97)	0.503 (92)	0.498 (97)	0.781 (91)	0.493 (96)	0.501 (92)	0.927 (49)	0.834 (73)	0.9298 (47)

Panel B: CNN

	K=3			K=5			K=7		
Num. Obs.	N=10	N=50	N=100	N=10	N=50	N=100	N=10	N=50	N=100
50	1.680 (18)	1.218 (18)	1.524 (19)	1.633 (6)	1.738 (5)	1.195 (27)	2.177 (17)	1.374 (7)	1.500 (24)
200	0.855 (42)	0.760 (47)	0.687 (65)	1.618 (6)	1.492 (7)	1.399 (12)	1.131 (21)	1.105 (20)	1.309 (11)
500	1.207 (11)	1.198 (10)	1.119 (11)	1.380 (9)	1.174 (23)	1.117 (24)	1.335 (9)	1.259 (12)	1.390 (8)

Panel C: CNN-LSTM

	K=3			K=5			K=7		
Num. Obs.	N=10	N=50	N=100	N=10	N=50	N=100	N=10	N=50	N=100
50	1.587 (17)	1.138 (19)	1.447 (17)	1.750 (13)	1.758 (12)	1.339 (18)	2.711 (8)	1.583 (5)	1.509 (13)
200	1.182 (38)	0.746 (57)	0.740 (55)	1.654 (5)	1.385 (5)	1.340 (13)	1.249 (10)	1.195 (18)	1.264 (9)
500	1.073 (21)	0.713 (63)	0.698 (69)	1.400 (12)	1.300 (19)	1.219 (25)	1.360 (7)	1.296 (12)	1.398 (8)

Note. The count of the Diebold-Mariano tests is depicted in brackets to right of the MSFE ratio. The Diebold-Mariano test is performed using the MSFEs of each individual data set, comparing the deep learning against the boosting model (H_a: The accuracy of the deep learning model is better than the boosting model) at a 10%-significance.

Out of the three deep learning models, the LSTM is found to perform the best. It shows a consistent outperformance of the boosting model, except when the number of observations in the simulation is 50, the number of variables is 50, and the number of underlying factors is 5 or 7. As can be expected, the model improves as the number of observations increases. This can be attributed to the fact that more observations can be used for estimation of the model parameters. On top of this, the model becomes better in relation to the boosting model when the number of observations becomes larger. However, as the number of underlying factors grows, the the MSFE ratio becomes larger if the number of observations and number of variables is kept the same. This means that the boosting model works better if more underlying factors are introduced into the time series. This is most probably due to the use of PCA to find these underlying factors. As such, the LSTM is best used in cases with many observations and many variables, but with a small amount of underlying factors. Due to the many intricate systems at play in macroeconomics and thus the likelihood of a big amount of underlying factors, the LSTM seems unsuitable for macroeconomic prediction.

Opposed to the LSTM model, the CNN and CNN-LSTM model perform much worse when compared to the boosting algorithm. Especially in cases with a low amount of observations, they fail to attain a higher accuracy than the boosting model. As the number of observations in the data set increased, the models perform better. The same holds for the number of variables: as the number of these increases, both the CNN and CNN-LSTM perform better. While both models perform best when used in cases with three underlying factors, they perform better with seven underlying factors than for cases with five. As such, while boosting is probably better in situations with clear underlying factors, it seems possible that the CNN and CNN-LSTM model perform better in situation with many underlying factors, such as the macroeconomic environment in this study.

5.2 Replication

Following the specifications described by Kim & Swanson (2018) for the AR and boosting model, I obtain MSFE ratios of 0.867, 0.864 and 0.870, for the one- ($h = 1$), three- ($h = 3$) and twelve-month-ahead ($h = 12$) forecasts respectively. This is very similar to the result of Kim & Swanson (2014b), who find the ratios to be equal to 0.871 and 0.867 for the one- and three-month-ahead prediction. The ratio for the twelve-month-ahead is unfortunately not available in the literature. As such, it is reasonable to assume that the methods have been implemented correctly. DM-tests with an alternative hypothesis of better accuracy give test statistics of 2.662, 3.068 and 2.580 for the three respective horizons. As such, I find that for all three horizons the null-hypothesis of equal predictive accuracy is rejected at $\alpha = 1\%$ in favor of the boosting model.

5.3 Deep Learning in Macroeconomic Forecasting

As deep learning models generally require a long estimation period to perform well, I use more of the available data for estimation of the models and less for forecast comparison, see Section 4.2 for further details on this. The performance of the best model for each architecture is presented in Table 4 below, along with the parameters that are used for the best model in question. Here, the best models were chosen based on the MSFE ratio after performing a grid search. The results for all the possible parameters mentioned in Section 4.3.3 can be found in Table 8 in Appendix C.

Table 4: MSFE ratios and hyperparameters of the best deep learning models compared to the benchmark AR and boosting models. *(E.g. for the entry 1.251 found under LSTM and then AR, this means that the LSTM is 25.1% worse than the AR model.)*

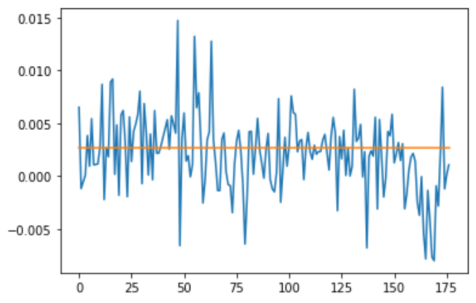
Horizon	Boosting	LSTM		**CNN**		CNN-LSTM	
		AR	Boosting	**AR**	**Boosting**	AR	Boosting
1	1.017	1.251**	1.230**	**1.027**	**1.010**	1.175**	1.156**
3	1.061	1.185*	1.118	**1.011**	**0.953**	1.118	1.054
12	1.185***	1.318***	1.112	1.156***	0.975	**1.127***	**0.951**
		Hyperparameters					
1		Num. Neurons: 32		Num. Filters 1: 32		Num. Filters 1: 32	
		Dropout Rate: 0.3		Num. Filters 2: 16		Num. Filters 2: 64	
3		Num. Neurons: 32		Num. Filters 1: 8		Num. Filters 1: 8	
		Dropout Rate: 0.3		Num. Filters 2: 64		Num. Filters 2: 16	
12		Num. Neurons: 16		Num. Filters 1: 32		Num. Filters 1: 4	
		Dropout Rate: 0.1		Num. Filters 2: 64		Num. Filters 2: 64	

Note. *p<0.1; **p<0.05; ***p<0.01; The best models were selected based on hyperparameter optimization, and for the convolutional neural network, any models that fell trap to a local minimum were removed prior to selection. A Diebold-Mariano test is performed using the MSFE, comparing the model against the AR and boosting model separately (H_a: Different from AR/Boosting). The best deep learning model for each forecast horizon is highlighted in bold.

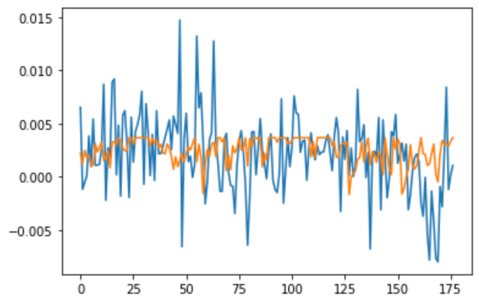
Interestingly, it is found that the boosting algorithm now has a ratio higher than one for all three the forecasting horizons (h = 1, 3, 12), with the twelve month horizon even showing significantly worse performance at the 1%-level. This can possibly be attributed to the longer estimation window, and much shorter window for forecast comparison. Another explanation could be that the GDP followed a more stable autoregressive pattern in later years than it did in earlier decades, which would greatly decrease the necessity of pre-selecting important macroeconomic variables through boosting and using PCA to find underlying factors. This possibly affected the performance ratio of boosting compared to the AR model. One other important notion is that the AR model used by Kim & Swanson (2014a,b, 2018) is only based on a small number of lags, which makes it difficult to forecast twelve steps ahead with this method. As a result, these forecasts can be found very close to a constant value throughout time. If boosting is then estimating several values incorrectly, while the AR model predicts close to the mean of the series, this can result in the AR model showing better performance, while this is not the case.

Looking at the deep learning models, the three models can be ranked based on their performance in comparison to the boosting and AR models. For the one-month and three-month forecasting horizons (h = 1, 3), I find that the convolutional neural networks (CNN) has the lowest ratios, when compared to both the AR and boosting models, than the other two deep learning models. For the twelve month forecast horizon (h = 12), the CNN-LSTM has the lowest ratios. That said, none of the deep learning models significantly outperforms the AR and boosting models and the ratios are all very close to one. A possible explanation is, again, the short horizon for which the forecasts are compared and, in the case of the twelve-month horizon, very constant predictions by the AR model might have caused improper results for this model. Based on the MSFE ratios and primarily focusing on the comparison of the deep learning models with boosting, the preferred model is CNN for the one-month- and three-month-ahead forecasts, while the CNN-LSTM combination is the preferred option for the twelve-month-ahead forecast.

Next, I will discuss the three models more in-depth, starting with the CNN, the best model out of the three. The CNN model has a ratio higher than one when compared to the autoregressive model for all three horizons, but it is only significantly outperformed for the twelve-month horizon, possibly due to reasons already explained above. That said, the difference between the CNN and AR forecasts for the one- and three-month-ahead horizons are very small at 2.76% and 1.1% respectively. This is considerably better than the other two models, which have accuracy differences of percentages upwards of 11.8%. Compared to the boosting model, the CNN model has a ratio slightly higher than one at 1.010 for the one-month forecasting horizon, while it is lower than one for the other two forecast horizons (h = 3 and 12). None of the three are significantly different based on the DM-test and therefore none can be said to perform better than boosting. While the CNN has the best performance out of the three models, it does come with a caveat: the model can sometimes fall trap to a local minimum for some of the parameters. This results in a constant value for all predictions or an upper ceiling in the predictions, of which examples found in the grid search can be seen in Figure 3 below. It could also be possible that lower floors are observed. As such, careful use of this model is recommended. The cause of this can be most likely attributed to a small learning rate. While a small learning rate allows for accurate estimation of the best parameters, the small deviations cannot force the model out of a local minimum. In this research, any of these cases where filtered out before selecting the best possible model for the CNN, but can be found in Table 8 in Appendix C. Both the LSTM and CNN-LSTM do not fall trap to this problem.



(a) Constant value predictions of a CNN model (orange) plotted along with the true value (blue) of GDP; CNN model for one-month-ahead forecasts, with respectively 4 and 16 filters for the two convolutional layers



(b) Predictions of a CNN model (orange) with a prediction ceiling, plotted along with the true value (blue) of GDP; CNN model for three-month-ahead forecasts, with respectively 32 and 16 filters for the two convolutional layers

Figure 3: Possible CNN prediction faults

The model with the second best comparison ratios for the one-month-ahead and three-month-ahead forecasts is the CNN-LSTM combination, while it is the best model for the twelve-month-ahead horizon. The model is significantly outperformed at the 5%-level by both the AR and boosting model for the one-month-ahead horizon. It also has a ratio higher than one for the three-month ahead forecasts, but is not significantly outperformed at this horizon. For the twelve-month-ahead forecasts, I find that the AR model significantly outperforms the CNN-LSTM model, but this is possibly due to the reasons surrounding the AR predictions that were already mentioned above. Meanwhile, the CNN-LSTM has a ratio lower than one when compared to the boosting algorithm for this horizon. While the ratio is only slightly lower than in the case of the regular CNN, and neither are significant, it does show that an additional LSTM component could be beneficial in cases where the horizon is larger. Future research could focus in on this model and look at much longer horizons beyond the current maximum of twelve months.

The last model to be discussed, the LSTM, is also the worst performing model based on the MSFE ratios. It is significantly outperformed by the AR model on all three horizons and by the boosting model for the one-month-ahead horizon. At the same time, it is 11.8% and 11.2% worse than the boosting model for the three-months- and twelve-month-ahead forecast horizons. This leads to the idea that, while LSTMs offer great value in forecasting time series, it fails to extract the key underlying factors that occur in such a big data macroeconomic set-up. That said, the depth of the model, with only one LSTM layer, might have also contributed to the poor results and lack of ability to find important underlying dependencies.

5.4 Variable Selection in Convolution Neural Networks

Given the predictive accuracy of the CNN, this model seems best at capturing underlying factors compared to the other two deep learning models. In addition to this, the convolutional layers used in this architecture allow for interesting insights in terms of the importance of different sections of the data. This stems from the filters that are applied to the data, for which the parameters are estimated such that they uncover the most important patterns in the data. The CNN architecture then learns which of these patterns carries the most value for the task at hand. In image recognition and classification, these filters learn to distinguish characteristics of the image, such as the snout of a dog or the ears of a cat, which is used to determine the type of image. Interestingly, the effect of these filters can be portrayed visually, where the importance of a data point is given by the brightness of the visual element. To again refer to the example of image recognition, one would for example be able to see the outline of the snout in a particular filter, while another filter outlines a cat's ears.

In time-series prediction, this offers the opportunity to look at which of the data is most important in predicting future values. This is similar to the notion of influential outliers in regression modeling, which are those observations that would significantly change the result if omitted. In finding which of the data is most important for the CNN, I focus on the first horizon and use the best found model. Before exploring the feature maps, I retrain this model with 100, instead of 50 epochs, to potentially increase accuracy. I then plot the values that the filters give to each data point for the first observation in the test set. This plot is provided in Figure 4 and also in Figure 5 in Appendix D. In Figure 4, the upper plot contains 32 filters with each 18 (half of the 36 months used as input to the model) values on the x-axis and 77 (half of the 144 input variables) values on y-axis. As such, a new filter starts each 18 horizontal steps. A similar construction holds for the second convolutional layer. The brighter the color, the higher the value that the filter gives to a particular data point.

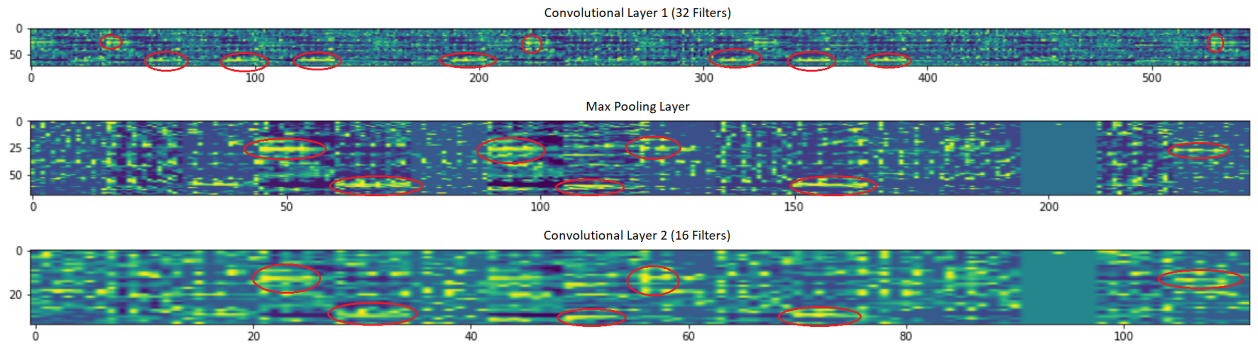


Figure 4: Feature maps of the best CNN model for a one-month-ahead ($h = 1$) forecast horizon. The plots represents the values provided to the data points by the filters in the first convolutional layer, the first max-pooling layer and the second convolutional layers. For the first convolutional layer, the plot contains 32 filters with each 18 (half of the 36 months used as input to the model) values on the x-axis and 77 (half of the 144 input variables) values on y-axis. As such, a new filter starts each 18 horizontal steps. These values are summarized in a max-pooling layer. The second convolutional layer contains 16 filters with 8 values each on the x-axis and 38 values on the y-axis. The brighter the data point is, the more value is attributed to this data point by the filter and subsequently the CNN. Important areas within the plots are highlighted with red circles.

First and foremost, Figure 4 shows that the convolutional network generally uses many sporadic parts of the variables and historic data points available. For some filters it even occurs that it is unable to find any patters. These filters are shown to be completely one color, without bright points. That said, Figure 4 does show that there are several patches of observations, see the parts circled in red, that are found to be important by multiple filters. These patches, especially the fact that they spread vertically, show that several variables are key in the prediction of GDP. Two groups of variables are found to be important by multiple filters. First, very bright patches are found at the bottom of some filters, see the bottom red circles in all three plots, which correspond approximately to the interest rate variables 127-134. The importance of these patches is furthermore strengthened by the fact that these areas are very dark in other filters, because, in the same way that high values carry a lot of value for the CNN, very low values have a similar impact. The second patch is found slightly higher in the three plots and also approximately corresponds to interest rate variables, this time the variables 53-59. While it is clear that interest rates thus offer considerable value for the CNN in predicting GDP of the United States. It is important to note that many of the other variables can be of considerable value, yet through smaller influences instead of large patches. When looking at the other three forecasting horizons, the same variables are found to be important. For these forecast horizons, the feature maps for respectively the one- and three-month-ahead forecasts are found in Figure 6 and 7 in Appendix D.

6 Conclusion

The initial goal of this research was to apply deep learning to macroeconomic forecasting of the United States (US) gross domestic product (GDP) in a big data setting and determine if it could outperform the boosting model by Bai & Ng (2008) and applied by Kim & Swanson (2018). This comparison included a simulation study, where the models were compared on data with clear underlying factors in multiple situations and the application of the deep learning models in the empirical situation of forecasting monthly US GDP.

In the simulation study, it was found that the long short-term memory (LSTM) neural network could outperform boosting in settings with a low number of clearly defined underlying factors and sufficient observations. However, when applying this model to the US macroeconomic data to predict US GDP, in which many intricate systems are at play, it failed to outperform the boosting model and even an autoregressive (AR) model. Oppositely, it was found that the convolutional neural network (CNN) and CNN-LSTM combination could not outperform boosting in the simulated environment but performed better than the LSTM in the empirical setting. More specifically, it was found in the empirical study that the three models do not significantly outperform the two benchmark models for a one-, three- and twelve-month-ahead horizon. Among the three models, especially the LSTM underperformed, as AR was significantly better at forecasting GDP for all three horizons, while the boosting algorithm also beat the LSTM model. The CNN-LSTM model proved to be the second best deep learning model for the one- and three-month-ahead forecasts and best for the twelve-month-ahead horizon. For the latter, it even proved better than the boosting algorithm, although not significantly. The best model for the one-month- and three-month-ahead forecasts is the CNN model, for which it came very close to the predictive accuracy of the AR and boosting models. It surpassed the boosting method for the three-months- and twelve-month-ahead forecasts. As a result, it shows potential for such big data time series forecasting. Concluding, the LSTM model seems best for situations with clearly defined underlying factors, which is generally not the case for macroeconomic systems, while the CNN and CNN-LSTM perform much better than the LSTM in more complicated environments. Here, the CNN-LSTM becomes more useful as the forecasting horizon increases.

As the best model, and due to its capability to display patterns found in the different layers, the CNN model was used to examine which variables were of greatest importance for predicting GDP. The most important variables for the CNN model were found to be mostly interest rate variables.

That said, much more research needs to go into understanding pattern recognition by convolutional layers, which could help determine the true importance of the different variables. Additionally, it is important to note that several other variables were also found to be important, but occured in much smaller quantities.

While interesting insights have been discovered, key limitations existed in this research. First, the use of several deep learning models limited the depth with which each model could be examined and optimized. As was seen during optimization, the performance depends heavily on the choice of model and hyperparameters. With more attention, these models could possibly perform better. Additionally, by only looking at one macroeconomic variable, it is unclear how deep learning models perform across macroeconomic forecasting as a whole. By including more of the dependent variables used by Kim & Swanson (2018), more can be uncovered about the performance of deep learning models.

Besides these limitations, this research has also uncovered a range of opportunities for future research to the applicability of deep learning in macroeconomic forecasting. First, and foremost, deep learning models offer a large range of set-ups and this research has found that hyperparameter optimization is of considerable importance. As such, different models can be further explored in the future, for which several options are: the ConvLSTM, a convolutional layer and lstm layer in one, a combination of one-dimensional convolutional layers for each separate variable, and the list goes on. While new models can be explored, particularly interesting is also the use of CNNs, as these have shown promising results for their first-time use in macroeconomic prediction on this scale. This warrants for interesting other opportunities for this method in time-series prediction, both in- and outside the field of macroeconomics. On top of this, these models offer a look at how the different variables add to the final prediction; in essence, which variables offer the most value.

This also leads me to the second option for future research: delving into the meaning of feature maps and the use of filters in pattern recognition for time series prediction. A lot is known about this for image recognition and classification, but it could add tremendous value to the use of CNNs if more is known about filters and their abilitiy to recognize patterns in a time-series domain. A third point for future research is to look into the combination of factor decomposition methods with deep learning. By first constructing the underlying factors of the data, the variable selection step is taken away from the deep learning models. The focus will then lie on finding the best predictive deep learning model. This research has shown that this could especially benefit the LSTM, which struggled with the large amount of variables.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Version 2.4.1)

Alaminos, D., Salas, M. B., & Fernández-Gámez, M. A. (2021). Quantum computing and deep learning methods for gdp growth forecasting. *Computational Economics*. doi: 10.1007/s10614-021-10110-z

Albawi, S., Abed Mohammed, T., & ALZAWI, S. (2017, 08). Understanding of a convolutional neural network.. doi: 10.1109/ICEngTechnol.2017.8308186

Arora, V., & Vamvakidis, A. (2006). The impact of u.s. economic growth on the rest of the world: How much does it matter. *Journal of Economic Integration*, *21*, 21-39.

Bai, J., & Ng, S. (2002, 1). Determining the number of factors in approximate factor models. *Econometrica*, *70*, 191-221. Retrieved from `http://doi.wiley.com/10.1111/1468-0262.00273` doi: 10.1111/1468-0262.00273

Bai, J., & Ng, S. (2008). Large dimensional factor analysis. *Foundations and Trends® in Econometrics*, *3*. doi: 10.1561/0800000002

Bai, J., & Ng, S. (2009, 6). Boosting diffusion indices. *Journal of Applied Econometrics*, *24*. doi: 10.1002/jae.1063

Bedi, J., & Toshniwal, D. (2019, 3). Deep learning framework to forecast electricity demand. *Applied Energy*, *238*. doi: 10.1016/j.apenergy.2019.01.113

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017, 3). Conditional time series forecasting with convolutional neural networks. Retrieved from `http://arxiv.org/abs/1703.04691`

Bühlmann, P., & Yu, B. (2003). Boosting with the l2 loss. *Journal of the American Statistical Association*, *98*(462), 324-339. doi: 10.1198/016214503000125

Cao, J., & Wang, J. (2019, 8). Stock price forecasting model based on modified convolution neural network and financial time series analysis. *International Journal of Communication Systems*, *32*. doi: 10.1002/dac.3987

Cook, T., & Hall, A. S. (2017). Macroeconomic indicator forecasting with deep neural networks. *The Federal Reserve Bank of Kansas City Research Working Papers*. doi: 10.18651/rwp2017-11

Diebold, F., & Mariano, R. (1995). Comparing predictive accuracy. *Journal of Business Economic Statistics*, *13*(3), 253-63. Retrieved from `https://EconPapers.repec.org/RePEc:bes:jnlbes:v:13:y:1995:i:3:p:253-63`

Economics, F. (2020, 10). *U.s. economic outlook.* Retrieved from `https://www.focus-economics.com/countries/united-states`

Fernald, J. G., & Jones, C. I. (2014, January). *The future of u.s. economic growth* (Working Paper No. 19830). National Bureau of Economic Research. Retrieved from `http://www.nber.org/papers/w19830` doi: 10.3386/w19830

Franses, P. H., & Dijk, D. v. (2000). *Non-linear time series models in empirical finance.* Cambridge University Press. doi: 10.1017/CBO9780511754067

Freund, Y., & Schapire, R. E. (1999). A brief introduction to boosting. In *Proceedings of the 16th international joint conference on artificial intelligence - volume 2* (p. 1401–1406). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The Annals of Statistics*, *28*(2), 337 – 407. Retrieved from `https://doi.org/10.1214/aos/1016218223` doi: 10.1214/aos/1016218223

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189 – 1232. Retrieved from `https://doi.org/10.1214/aos/1013203451` doi: 10.1214/aos/1013203451

Georgiadis, G. (2016, 10). Determinants of global spillovers from us monetary policy. *Journal of International Money and Finance*, *67*, 41-61. doi: 10.1016/j.jimonfin.2015.06.010

Groen, J. J., & Kapetanios, G. (2016, 8). Revisiting useful approaches to data-rich macroeconomic forecasting. *Computational Statistics and Data Analysis*, *100*, 221-239. doi: 10.1016/j.csda.2015.11.014

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735

Jin, X., Yu, X., Wang, X., Bai, Y., Su, T., & Kong, J. (2020). *Prediction for time series with cnn and lstm.* doi: 10.1007/978-981-15-0474-7_59

Kim, H. H., & Swanson, N. R. (2014a, 1). Forecasting financial and macroeconomic variables using data reduction methods: New empirical evidence. *Journal of Econometrics*, *178*. doi: 10.1016/j.jeconom.2013.08.033

Kim, H. H., & Swanson, N. R. (2014b). *Mining big data using parsimonious factor and shrinkagemethods.* Kim and Swanson.

Kim, H. H., & Swanson, N. R. (2018). Mining big data using parsimonious factor, machine learning, variable selection and shrinkage methods. *International Journal of Forecasting*, *34*. doi: 10.1016/j.ijforecast.2016.02.012

Koenecke, A. (2020). *Applying deep neural networks to financial time series forecasting.*

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*. doi: 10.1109/5.726791

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018, 04). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18*, 1-52.

Liu, S., Ji, H., & Wang, M. C. (2020, 8). Nonpooling convolutional neural network forecasting for seasonal time series with trends. *IEEE Transactions on Neural Networks and Learning Systems*, *31*. doi: 10.1109/TNNLS.2019.2934110

Livieris, I. E., Pintelas, E., & Pintelas, P. (2020, 12). A cnn–lstm model for gold price time-series forecasting. *Neural Computing and Applications*, *32*, 17351-17360. doi: 10.1007/s00521-020-04867-x

Maćkowiak, B. (2007, 11). External shocks, u.s. monetary policy and macroeconomic fluctuations in emerging markets. *Journal of Monetary Economics*, *54*, 2512-2520. doi: 10.1016/j.jmoneco.2007.06.021

Miranda-Agrippino, S., & Rey, H. (2020). U.s. monetary policy and the global financial cycle. *Review of Economic Studies*, *87*, 2754-2776. Retrieved from `https://academic.oup.com/restud/article/87/6/2754/5834728` doi: 10.1093/restud/rdaa019

Nelson, M., Hill, T., Remus, W., & O'Connor, M. (1999). Time series forecasting using neural networks: Should the data be deseasonalized first? *Journal of forecasting*, *18*(5), 359–367.

Neuenkirch, M., & Neumeier, F. (2015). The impact of un and us economic sanctions on gdp growth. *European Journal of Political Economy*, *40*, 110-125. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0176268015000816` doi: https://doi.org/10.1016/j.ejpoleco.2015.09.001

Obstfeld, M. (2019, 7). *Global dimensions of u.s. monetary policy.* Retrieved from `http://www.nber.org/papers/w26039.pdf` doi: 10.3386/w26039

Olah, C. (2015). Understanding lstm networks. Retrieved from `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. (Version 0.24)

Qing, X., & Niu, Y. (2018, 4). Hourly day-ahead solar irradiance prediction using weather forecasts by lstm. *Energy*, *148*. doi: 10.1016/j.energy.2018.01.177

Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. In *9th python in science conference.* Retrieved from `https://www.statsmodels.org/stable/index.html` (Version 0.12.2)

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017, 11). Stock price prediction using lstm, rnn and cnn-sliding window model. In (Vol. 2017-January, p. 1643-1647). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICACCI.2017.8126078

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020, 5). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing Journal*, *90*, 106181. doi: 10.1016/j.asoc.2020.106181

Stock, J., & Watson, M. (2002a, 12). Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, *97*, 1167-1179. doi: 10.1198/016214502388618960

Stock, J., & Watson, M. (2002b). Macroeconomic forecasting using diffusion indexes. *Journal of Business and Economic Statistics*, *20*(2), 147–162.

Stock, J., & Watson, M. (2012, 10). Generalized shrinkage methods for forecasting using many predictors. *Journal of Business Economic Statistics*, *30*. doi: 10.1080/07350015.2012.715956

Theoharidis, A. F. (2021). *Forecasting inflation using deep learning: An application of convolutional lstm networks and variational autoencoders .*

Yang, J.-B., Nhut, N., San, P., li, X., & Shonali, P. (2015, 07). Deep convolutional neural networks on multichannel time series for human activity recognition. *IJCAI*.

Yang, X., & Liu, X. (2021). *A deep hybrid neural network forecasting for multivariate non-stationary time series.* doi: 10.1007/978-3-030-70626-5_19

Zhao, B., Lu, H., Chen, S., Liu, J., & Wu, D. (2017, 2). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, *28*. doi: 10.21629/JSEE.2017.01.18

Zhou, D.-X. (2018, 11). Deep distributed convolutional neural networks: Universality. *Analysis and Applications*, *16*. doi: 10.1142/S0219530518500124

Appendix A. List of Variables used in Empirical Study

Table 5 below contains a description of the used variables in the empirical study, along with their transformations. The transformation codes (TCodes) correspond to the following transformations:

1. Level of the series
2. First difference of the series
3. Second difference of the series
4. Log of the series
5. First difference of the log of the series
6. Second difference of the log of the series

Finally, the dependent variable in this research is highlighted in bold and corresponds to variable number 144.

Table 5: List of all variables used in this research, including description and the transformation used during preprocessing of the data

Num.	**Short**	**Long Description**	**TCodes**
1	A0M007	Manufacturer's New Orders, Durable Goods Industries (Bil. Chain 2000$)	5
2	A0M051	Personal Income Less Transfer Payments (Ar, Bil. Chain 2000 $)	5
3	CCINRV	Consumer Credit Outstanding - Nonrevolving(g19)	5
4	CCIPY	Consumer Instal Credit to Personal Income, Ratio (%,Sa)(bcd-95)	5
5	CES001	Employees on Nonfarm Payrolls - Total Nonfarm	5
6	CES002	Employees on Nonfarm Payrolls - Total Private	5
7	CES003	Employees on Nonfarm Payrolls - Goods-Producing	5
8	CES006	Employees on Nonfarm Payrolls - Mining	5
9	CES011	Employees on Nonfarm Payrolls - Construction	5
10	CES015	Employees on Nonfarm Payrolls - Manufacturing	5
11	CES017	Employees on Nonfarm Payrolls - Durable Goods	5
12	CES033	Employees on Nonfarm Payrolls - Nondurable Goods	5
13	CES046	Employees on Nonfarm Payrolls - Service-Providing	5
14	CES048	Employees on Nonfarm Payrolls - Trade, Transportation, and Utilities	5
15	CES049	Employees on Nonfarm Payrolls - Wholesale Trade	5
16	CES053	Employees on Nonfarm Payrolls - Retail Trade	5
17	CES088	Employees on Nonfarm Payrolls - Financial Activities	5
18	CES128	Employees on Nonfarm Payrolls - Leisure and Hospitality	5
19	CES140	Employees on Nonfarm Payrolls - Government	5

Table 5: List of all variables used in this research, including description and the transformation used during preprocessing of the data

Num.	Short	Long Description	TCodes
20	CES151	Average Weekly Hours of Production or Nonsupervisory Workers on Private Nonfarm Payrolls - Goods-Producing	2
21	CES155	Average Weekly Hours of Production or Nonsupervisory Workers on Private Nonfarm Payrolls - Manufacturing Overtime Hours	2
22	CES156	Average Weekly Hours of Production or Nonsupervisory Workers on Private Nonfarm Payrolls - Durable Goods	2
23	CES275	Average Hourly Earnings of Production or Nonsupervisory Workers on Private Nonfarm Payrolls - Goods	5
24	CES276	Average Hourly Earnings of Production or Nonsupervisory Workers on Private Nonfarm Payrolls - Natural Resources and Mining Current Dollars	5
25	CES277	Average Hourly Earnings of Production or Nonsupervisory Workers on Private Nonfarm Payrolls - Construction	5
26	CES278	Average Hourly Earnings of Production or Nonsupervisory Workers on Private Nonfarm Payrolls - Manufacturing	5
27	CES295	Indexes of Aggregate Weekly Hours of Production - Goods-Producing	5
28	CES297	Indexes of Aggregate Weekly Hours of Production - Construction	5
29	CES298	Indexes of Aggregate Weekly Hours of Production - Manufacturing	5
30	CES299	Indexes of Aggregate Weekly Hours of Production - Durable Goods	5
31	CES310	Indexes of Aggregate Weekly Hours of Production - Nondurable Goods	5
32	EXRCAN	Foreign Exchange Rate: Canada (Canadian $ per U.s.$)	5
33	EXRJAN	Foreign Exchange Rate: Japan (Yen per U.s.$)	5
34	EXRSW	Foreign Exchange Rate: Switzerland (Swiss Franc per U.s.$)	5
35	EXRUK	Foreign Exchange Rate: United Kingdom (Cents per Pound)	5
36	EXRUS	United States;effective Exchange Rate(merm)(Index No.)	5
37	FCLIN	Loans & Sec @ All Coml Banks: Loans to Individuals (Bil$, sa)	5
38	FCLRE	Loans & Sec @ All Coml Banks: Real Estate Loans (Bil$, sa)	5
39	FCLS	Loans & Sec @ All Coml Banks: Total (Bils, sa)	5
40	FCSGV	Loans & Sec @ All Coml Banks: U.s. Govt Securities (Bil$, sa)	5
41	FM1	Money Stock: M1(curr,trav.cks,dem Dep,other Checkable Dep)(bil$,sa)	5
42	FM2	Money Stock:m2(m1+overnite Rps,euro$,g/p&b/d Mmmfs&sav&sm Time Dep(bil$)	5

Table 5: List of all variables used in this research, including description and the transformation used during preprocessing of the data

Num.	**Short**	**Long Description**	**TCodes**
43	FMFBA	Monetary Base, Adj for Reserve Requirement Changes(mil$,sa)	5
44	FMRRA	Depository Inst Reserves:total,adj for Reserve Req Chgs(mil$,sa)	5
45	FSDXP	S&p's Composite Common Stock: Dividend Yield (% per Annum)	2
46	FSPCOM	S&p's Common Stock Price Index: Composite (1941-43=10)	5
47	FSPIN	S&p's Common Stock Price Index: Industrials (1941-43=10)	5
48	FSPXE	S&p's Composite Common Stock: Price-Earnings Ratio (%,nsa)	5
49	FSTE	U.S. MDSE Exports: Total Exports (F.A.S. Value) (Mil.$, sa)	5
50	FYAAAC	Bond Yield: Moody's Aaa Corporate (% per Annum)	2
51	FYAC	Bond Yield: Moody's a Corporate (% per Annum)	2
52	FYBAAC	Bond Yield: Moody's Baa Corporate (% per Annum)	2
53	FYFF	Interest Rate: Federal Funds (Effective) (% per Annum,nsa)	2
54	FYGM3	Interest Rate: U.S.treasury Bills,sec Mkt,3-Mo.(% per Ann,nsa)	2
55	FYGM6	Interest Rate: U.S.treasury Bills,sec Mkt,6-Mo.(% per Ann,nsa)	2
56	FYGT1	Interest Rate: U.S.treasury Const Maturities,1-Yr.(% per Ann,nsa)	2
57	FYGT10	Interest Rate: U.S.treasury Const Maturities,10-Yr.(% per Ann,nsa)	2
58	FYGT3	Interest Rate: U.S.treasury Const Maturities,3-Yr.(% per Ann,nsa)	2
59	FYGT5	Interest Rate: U.S.treasury Const Maturities,5-Yr.(% per Ann,nsa)	2
60	HHSNTN	U. of Mich. Index of Consumer Expectations(bcd-83)	2
61	HMOB	Mobile Homes: Manufacturers Shipments (thous. of Units, saar)	5
62	HSBMW	Houses Authorized by Build. Permits:midwest(thous.u., sa)	4
63	HSBNE	Houses Authorized by Build. Permits:northeast(thous.u., sa)	4
64	HSBR	Housing Authorized: Total New Priv Housing Units (thous.,saar)	4
65	HSBSOU	Houses Authorized by Build. Permits:south(thous.u.)s.a.	4
66	HSBWST	Houses Authorized by Build. Permits:west(thous.u.)s.a.	4
67	HSFR	Housing Starts:nonfarm(1947-58); total Farm&nonfarm(1959-)(thous.)s.a.	4
68	HSMW	Housing Starts:midwest(thous.u.)s.a.	4
69	HSNE	Housing Starts:northeast (thous.u.)s.a.	4
70	HSSOU	Housing Starts:south (thous.u.)s.a.	4
71	HSWST	Housing Starts:west (thous.u.)s.a.	4
72	IPS10	Industrial Production Index - Total Index	5
73	IPS11	Industrial Production Index - Products, Total	5
74	IPS12	Industrial Production Index - Consumer Goods	5
75	IPS13	Industrial Production Index - Durable Consumer Goods	5

Table 5: List of all variables used in this research, including description and the transformation used during preprocessing of the data

Num.	**Short**	**Long Description**	**TCodes**
76	IPS14	Industrial Production Index - Automotive Products	5
77	IPS18	Industrial Production Index - Nondurable Consumer Goods	5
78	IPS22	Industrial Production Index - Chemical Products	5
79	IPS25	Industrial Production Index - Business Equipment	5
80	IPS25	Industrial Production Index - Business Equipment	5
81	IPS26	Industrial Production Index - Transit Equipment	5
82	IPS299	Industrial Production Index - Final Products	5
83	IPS306	Industrial Production Index - Fuels	5
84	IPS307	Industrial Production Index - Residential Utilities	5
85	IPS31	Industrial Production Index - Business Supplies	5
86	IPS32	Industrial Production Index - Materials	5
87	IPS34	Industrial Production Index - Durable Goods Materials	5
88	IPS38	Industrial Production Index - Nondurable Goods Materials	5
89	IPS43	Industrial Production Index - Manufacturing (SIC)	5
90	LHEL	Index of Help-Wanted Advertising in Newspapers (1967=100;sa)	2
91	LHELX	Employment: Ratio; Help-Wanted Ads:no. Unemployed Clf	2
92	LHEM	Civilian Labor Force: Employed, Total (thous.,sa)	5
93	LHNAG	Civilian Labor Force: Employed, Nonagric.industries (thous.,sa)	5
94	LHU14	Unemploy.by Duration: Persons Unempl.5 to 14 Wks (thous.,sa)	5
95	LHU15	Unemploy.by Duration: Persons Unempl.15 Wks + (thous.,sa)	5
96	LHU26	Unemploy.by Duration: Persons Unempl.15 to 26 Wks (thous.,sa)	5
97	LHU27	Unemploy.by Duration: Persons Unempl.27 Wks + (Thous,sa)	5
98	LHU5	Unemploy.by Duration: Persons Unempl.less Than 5 Wks (thous.,sa)	5
99	LHU680	Unemploy.by Duration: Average(mean)duration in Weeks (Sa)	2
100	LHUR	Unemployment Rate: All Workers, 16 Years & Over (%,Sa)	2
101	MOCMQ	New Orders (Net)-Consumer Goods & Materials, 1992 Dollars (BCI)	5
102	MSONDQ	New Orders, Nondefense Capital Goods, in 1992 Dollars (BCI)	5
103	PMCP	Napm Commodity Prices Index (Percent)	2
104	PMDEL	Napm Vendor Deliveries Index (Percent)	2
105	PMEMP	Napm Employment Index (Percent)	2
106	PMI	Purchasing Managers' Index (Sa)	2
107	PMNO	Napm New Orders Index (Percent)	2
108	PMNV	Napm Inventories Index (Percent)	2
109	PMP	Napm Production Index (Percent)	2

Table 5: List of all variables used in this research, including description and the transformation used during preprocessing of the data

Num.	Short	Long Description	TCodes
110	PSCCOM	Spot Market Price Index:bls & Crb: All Commodities(1967=100)	5
111	PU45	Cpi-U: New Cars (82-84=100,sa)	5
112	PU83	Cpi-U: Apparel & Upkeep (82-84=100,sa)	5
113	PU84	Cpi-U: Transportation (82-84=100,sa)	5
114	PU85	Cpi-U: Medical Care (82-84=100,sa)	5
115	PUC	Cpi-U: Commodities (82-84=100,sa)	5
116	PUCD	Cpi-U: Durables (82-84=100,sa)	5
117	PUNEW	Cpi-U: All Items (82-84=100,sa)	5
118	PUS	Cpi-U: Services (82-84=100,sa)	5
119	PUXE	Cpi-U: All Items Less Energy (82-84=100,sa)	5
120	PUXF	Cpi-U: All Items Less Food (82-84=100,sa)	5
121	PUXHS	Cpi-U: All Items Less Shelter (82-84=100,sa)	5
122	PUXM	Cpi-U: All Items Less Medical Care (82-84=100,sa)	5
123	PWCMSA	Producer Price Index:crude Materials (82=100,sa)	5
124	PWFCSA	Producer Price Index:finished Consumer Goods (82=100,sa)	5
125	PWFSA	Producer Price Index: Finished Goods (82=100,sa)	5
126	PWIMSA	Producer Price Index:intermed Mat.supplies & Components(82=100,sa)	5
127	SFYAAAC	FYAAAC-FYFF	1
128	SFYBAAC	FYBAAC-FYFF	1
129	SFYBAC	FYAC-FYFF	1
130	SFYGM3	FYGM3-FYFF FYGM6-FYFF FYGT1-FYFF	1
131	SFYGM6	FYGM6-FYFF	1
132	SFYGT1	FYGT1-FYFF	1
133	SFYGT10	FYGT10-FYFF	1
134	SFYGT5	FYGT5-FYFF	1
135	UTL11	Capacity Utilization - Manufacturing (SIC),percent of Capacity, Sa, Frb	2
136	UTL15	Capacity Utilization - Nonmetallic Mineral Product Naics=327, Percent of Capacity, Sa, Frb	2
137	UTL17	Capacity Utilization - Fabricated Metal Product Naics=332, Percent of Capacity, Sa, Frb	2
138	UTL21	Capacity Utilization - Motor Vehicles and Parts Naics=3361-3, Percent of Capacity, Sa, Frb	2

Table 5: List of all variables used in this research, including description and the transformation used during preprocessing of the data

Num.	**Short**	**Long Description**	**TCodes**
139	UTL22	Capacity Utilization - Aerospace and Miscellaneous Transportation Eq., Percent of Capacity, Sa, Frb	2
140	UTL29	Capacity Utilization - Paper Naics=322, Percent of Capacity, Sa, Frb	2
141	UTL31	Capacity Utilization - Petroleum and Coal Products Naics=324	2
142	UTL32	Capacity Utilization - Chemical Naics=325	2
143	UTL33	Capacity Utilization - Plastics and Rubber Products Naics=326	2
144	GDP	**Gross Domestic Product Extrapolated Under CPI**	5

Appendix B. Supplementary Results Simulation Study

Table 6 contains the hyperparameters which were found to be the best parameters based on Hyperband optimization. Furthermore, Table 7 contains the results of the simulation study, including the average ratio of each data set for each of the simulation scenario's and a count of the number of DM-tests that showed significance outperformance of the deep learning model compared to the benchmark model. The DM-tests were based on a 10% significance level.

Table 6: Best parameters found using Hyperband optimization and used in the simulation study to estimate the model and forecast over the test set. *(E.g. the value 240, found in the fourth row and third column. means that the best model contained 240 recurrent components for the simulation scenario with 50 observations, 10 variables and 3 underlying factors.)*

Panel A: LSTM										
		K=3			K=5			K=7		
Num. Obs.	Parameter	N=10	N=50	N=100	N=10	N=50	N=100	N=10	N=50	N=100
50	Num. Neurons	240	192	208	192	240	240	240	192	208
	Dropout Rate	0.25	0.10	0.20	0.20	0.10	0.20	0.15	0.10	0.25
200	Num. Neurons	128	224	144	160	192	176	240	176	28
	Dropout Rate	0.30	0.30	0.25	0.20	0.30	0.20	0.25	0.20	0.25
500	Num. Neurons	192	144	176	176	192	240	208	224	192
	Dropout Rate	0.10	0.25	0.25	0.10	0.20	0.10	0.25	0.25	0,20
Panel B: CNN										
		K=3			K=5			K=7		
Num. Obs.	Parameter	N=10	N=50	N=100	N=10	N=50	N=100	N=10	N=50	N=100
50	Num. Filters 1	32	32	8	32	64	32	4	16	64
	Num. Filters 2	64	16	64	64	64	32	64	64	32
200	Num. Filters 1	64	8	64	32	32	16	8	16	32
	Num. Filters 2	64	32	16	16	8	8	64	32	64
500	Num. Filters 1	16	32	8	8	64	16	64	8	64
	Num. Filters 2	16	32	4	64	4	32	32	16	16
Panel C: CNN-LSTM										
		K=3			K=5			K=7		
Num. Obs.	Parameter	N=10	N=50	N=100	N=10	N=50	N=100	N=10	N=50	N=100
50	Num. Filters 1	64	32	32	64	4	16	16	32	64
	Num. Filters 2	64	16	32	64	64	64	64	32	64
200	Num. Filters 1	64	4	4	64	64	8	64	16	64
	Num. Filters 2	64	64	64	32	32	8	32	32	64
500	Num. Filters 1	16	32	8	8	64	64	32	8	64
	Num. Filters 2	16	32	32	64	16	4	64	16	4

Note. The best model for each horizon and model is highlighted in bold. In the table, N corresponds to the number of variables in the simulation data, K corresponds to the number of underlying factors and Num. Obs. refers to the number of observations T in the data sets.

Table 7: Average MSFE ratios and number of DM-tests that showed significantly better performance for the deep learning models compared to AR and boosting for the simulation scenarios. *(E.g. 0.598 in the 3rd column, and 5th row corresponds to a 40.2% better accuracy for the LSTM model compared to the boosting model, for the simulation scenario with $K = 3$ underlying factors, $N = 10$ variables and $T = 50$ number of observations.)*

Panel A: LSTM

	K=3						K=5						K=7					
	N=10		N=50		N=100		N=10		N=50		N=100		N=10		N=50		N=100	
Num. Obs.	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting
50	0.689	0.598	0.615	0.468	0.614	0.464	1.249	1.136	0.885	0.809	0.719	0.631	1.657	1.531	0.965	0.788	0.996	0.808
	(38)	(54)	(35)	(53)	(31)	(59)	(13)	(22)	(23)	(29)	(33)	(53)	(5)	(13)	(21)	(24)	(0)	(3)
200	0.604	0.522	0.917	0.444	0.509	0.418	0.912	0.831	0.676	0.603	0.593	0.559	0.823	0.784	0.634	0.591	0.591	0.576
	(89)	(93)	(94)	(100)	(95)	(98)	(32)	(39)	(73)	(85)	(94)	(97)	(51)	(57)	(93)	(95)	(97)	(99)
500	0.529	0.469	0.514	0.503	0.534	0.498	0.8145	0.780	0.547	0.493	0.552	0.501	0.955	0.927	0.872	0.834	0.952	0.929
	(96)	(97)	(89)	(92)	(93)	(97)	(86)	(91)	(95)	(96)	(93)	(92)	(43)	(49)	(63)	(73)	(45)	(47)

Panel B: CNN

	K=3						K=5						K=7					
	N=10		N=50		N=100		N=10		N=50		N=100		N=10		N=50		N=100	
Num. Obs.	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting
50	1.935	1.680	1.602	1.218	2.022	1.524	1.79602	1.633484	1.902027	1.737654	1.361331	1.194715	2.355088	2.176892	1.682443	1.374065	1.852479	1.50049
	(9)	(18)	(8)	(18)	(11)	(19)	(2)	(6)	(3)	(5)	(21)	(27)	(5)	(17)	(3)	(7)	(13)	(24)
200	0.989	0.855	0.892	0.760	0.838	0.687	1.774094	1.618143	1.672662	1.492298	1.798	1.533	1.186837	1.130816	1.172	1.105	1.351	1.309
	(28)	(42)	(25)	(47)	(29)	(65)	(3)	(6)	(5)	(7)	(4)	(5)	(12)	(21)	(15)	(20)	(9)	(11)
500	1.363	1.207	1.244	1.198	1.168	1.119	1.4405	1.38045	1.302049	1.173969	1.276	1.117	1.376319	1.33536	1.302	1.259	1.421	1.390
	(6)	(11)	(9)	(10)	(13)	(11)	(8)	(9)	(17)	(23)	(21)	(24)	(7)	(9)	(11)	(12)	(5)	(8)

Panel C: CNN-LSTM

	K=3						K=5						K=7					
	N=10		N=50		N=100		N=10		N=50		N=100		N=10		N=50		N=100	
Num. Obs.	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting
50	1.828	1.587	1.496	1.138	1.920	1.447	1.923	1.749	1.923	1.757	1.526	1.339	2.932	2.710	1.938	1.583	1.863	1.509
	(10)	(17)	(13)	(19)	(9)	(17)	(6)	(13)	(11)	(12)	(7)	(18)	(2)	(8)	(4)	(5)	(6)	(13)
200	1.326	1.182	0.877	0.746	0.903	0.740	1.813	1.654	1.551	1.384	1.89	1.772	1.311	1.249	1.212	1.195	1.302	1.264
	(21)	(38)	(33)	(57)	(31)	(55)	(2)	(5)	(5)	(5)	(3)	(4)	(7)	(10)	(13)	(18)	(6)	(9)
500	1.211	1.073	0.834	0.713	0.855	0.698	1.461	1.400	1.441	1.299	1.453	1.314	1.401	1.360	1.343	1.296	1.4124	1.398
	(9)	(21)	(51)	(63)	(60)	(69)	(9)	(12)	(13)	(19)	(10)	(18)	(6)	(7)	(11)	(12)	(5)	(8)

Note. The count of the Diebold-Mariano tests is depicted in brackets below of the corresponding MSFE ratio. The Diebold-Mariano test is performed using the MSFEs of each individual data set, comparing the deep learning against the AR or boosting model separately (H_a: The accuracy of the deep learning model is better than the benchmark model) at a 10%-significance.

Appendix C. Supplementary Results Empirical Study

Table 8 on the next page contains the ratio's of the mean square forecasting errors (MSFEs) of the deep learning models and the benchmark models. For example, the ratio 1.329 in column 3, row 4 of the table compares the LSTM model to an autoregressive (AR) model. This is given by

$$Ratio = \frac{MFSE_{LSTM}}{MFSE_{AR}}, \tag{22}$$

for the LSTM model with 16 recurrent components and a dropout rate of 0.1 at a one-month-ahead forecast horizon. The three deep learning are each represented in a separate panel and compared to both an AR model and the boosting algorithm used in this research. For this model, see Section 3.3.

Several of the convolutional neural networks fell trap to a local minimum or failed to properly model the time-series. As a result, these models only forecasted a single constant or experienced a ceiling in the forecasts, above which no predictions could be made. These models were excluded before selecting the best model for comparison in the results section of this research and are highlighted by a ^{1} in Table 8.

Table 8: MSFE ratios of the deep learning models compared to the benchmark AR and boosting models. *(E.g. the entry 1.329 found in the third column and fourth row, means that the LSTM with 16 neurons and a dropout rate of 0.1 is 32.9% worse than an AR model for predicting one-month-ahead forecasts.)*

Panel A: LSTM											
	$D\backslash N$	16		32		64		128		256	
Horizon		AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting
1	0.1	1.329	1.307	1.294	1.272	1.274	1.253	1.394	1.371	1.433	1.410
	0.2	1.253	1.232	1.357	1.334	1.278	1.256	1.382	1.359	1.512	1.487
	0.3	1.291	1.269	**1.251**	**1.230**	1.275	1.253	1.356	1.333	1.455	1.431
3	0.1	1.749	1.648	1.289	1.215	1.343	1.266	1.402	1.321	1.459	1.375
	0.2	1.547	1.458	1.398	1.318	1.405	1.325	1.288	1.215	1.454	1.371
	0.3	1.388	1.309	**1.185**	**1.118**	1.390	1.310	1.440	1.358	1.491	1.406
12	0.1	**1.318**	**1.112**	1.500	1.266	1.426	1.203	1.601	1.351	1.791	1.512
	0.2	1.433	1.209	1.478	1.247	1.401	1.183	1.534	1.295	1.781	1.503
	0.3	1.318	1.113	1.527	1.288	1.339	1.130	1.581	1.335	1.782	1.504
Panel B: CNN											
	$F2\backslash F1$	4		8		16		32		64	
Horizon		AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting
1	4	2.094	2.059	1.508	1.483	1.833	1.803	1.973	1.940	2.509	2.467
	8	1.170	1.151	1.711	1.682	1.265	1.244	1.385	1.362	1.850	1.819
	16	0.999[1]	0.982[1]	1.315	1.293	1.223	1.203	**1.027**	**1.010**	1.312	1.290
	32	1.254	1.233	0.998[1]	0.982[1]	1.144	1.125	0.998[1]	0.982[1]	1.633	1.606
	64	1.358	1.336	1.406	1.383	1.216	1.196	1.406	1.383	1.814	1.784
3	4	1.653	1.558	1.190	1.122	1.837	1.732	2.238	2.110	1.839	1.734
	8	1.370	1.292	1.328	1.252	1.443	1.360	1.710	1.612	1.532	1.445
	16	1.090	1.028	1.548	1.459	1.336	1.260	1.015	0.957	1.569	1.479
	32	1.011[1]	0.953[1]	1.433	1.351	1.010[1]	0.952[1]	1.013[1]	0.955[1]	1.456	1.373
	64	1.105	1.041	**1.011**	**0.953**	1.347	1.270	1.211	1.141	1.266	1.194
12	4	1.489	1.257	1.421	1.199	1.942	1.639	1.528	1.290	2.280	1.924
	8	1.729	1.459	1.677	1.415	1.734	1.463	1.484	1.252	1.720	1.452
	16	1.375	1.160	1.484	1.252	1.429	1.206	1.625	1.372	1.431	1.208
	32	1.477	1.247	1.468	1.239	0.996[1]	0.841[1]	1.332	1.124	0.997[1]	0.842[1]
	64	0.995[1]	0.840[1]	0.995[1]	0.840[1]	1.472	1.242	**1.156**	**0.975**	2.233	1.884
Panel C: CNN-LSTM											
	$F2\backslash F1$	4		8		16		32		64	
Horizon		AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting	AR	Boosting
1	4	1.967	1.934	1.857	1.826	1.419	1.396	1.978	1.945	1.646	1.618
	8	1.655	1.628	1.528	1.503	1.398	1.375	1.397	1.374	1.197	1.178
	16	1.510	1.485	1.422	1.398	1.316	1.294	1.284	1.263	1.459	1.435
	32	1.203	1.183	1.423	1.399	1.300	1.279	1.383	1.360	1.196	1.176
	64	1.222	1.201	1.412	1.388	1.365	1.343	**1.175**	**1.156**	1.333	1.311
3	4	1.769	1.668	2.235	2.107	1.612	1.520	1.899	1.790	1.654	1.560
	8	1.563	1.473	1.583	1.492	1.417	1.336	1.398	1.318	1.591	1.499
	16	1.412	1.331	**1.118**	**1.054**	1.492	1.407	1.216	1.146	1.396	1.316
	32	1.219	1.150	1.212	1.143	1.261	1.189	1.336	1.260	1.358	1.281
	64	1.197	1.128	1.323	1.247	1.281	1.208	1.288	1.214	1.382	1.302
12	4	1.919	1.619	1.563	1.319	1.402	1.183	1.989	1.678	1.663	1.404
	8	1.581	1.334	1.805	1.523	1.818	1.534	1.497	1.264	1.884	1.590
	16	1.392	1.175	1.338	1.130	1.375	1.160	1.492	1.259	1.886	1.592
	32	1.250	1.055	1.285	1.084	1.266	1.068	1.335	1.127	1.546	1.305
	64	**1.127**	**0.951**	1.386	1.170	1.362	1.149	1.392	1.174	1.511	1.276

Note. The best model for each horizon and model is highlighted in bold. In the table, N and d respectively correspond to the number of neurons and dropout rate of the LSTM layer and. $F1$ and $F2$ to the filters in respectively the first and second layer of the CNN and CNN-LSTM.

[1] Cases where the CNN got stuck in a local minimum and therefore only predicted a constant line or the predictions experienced a ceiling above which no predictions were made. These cases were not taken into account when selecting the best model.

Appendix D. Feature Maps of the Convolutional Neural Network

Figure 5, 6 and 7 below represent the importance that the filters of a convolutional neural networks (CNN) give to the different data points of an observation, which is also commonly referred to as feature maps. For more information about feature maps, see Section 3.4. In this case, the first observation of the test set is used in the visualization. Figure 5 contains the feature maps based on the best model for one-month-ahead forecast, Figure 6 those for three-month-ahead forecasts and Figure 7 those for twelve-month-ahead forecasts.

To illustrate the idea behind these plots, in Figure 5 the upper plot contains 32 filters with each 18 (half of the 36 months used as input to the model) values on the x-axis and 77 (half of the 144 input variables) values on y-axis. As such, a new filter starts each 18 horizontal steps. A similar construction holds for the second convolutional layer, which can be seen in the bottom plot of Figure 5. The brighter the color, the higher the value that the filter gives to a particular data point. Important areas within the plots are highlighted with red circles.

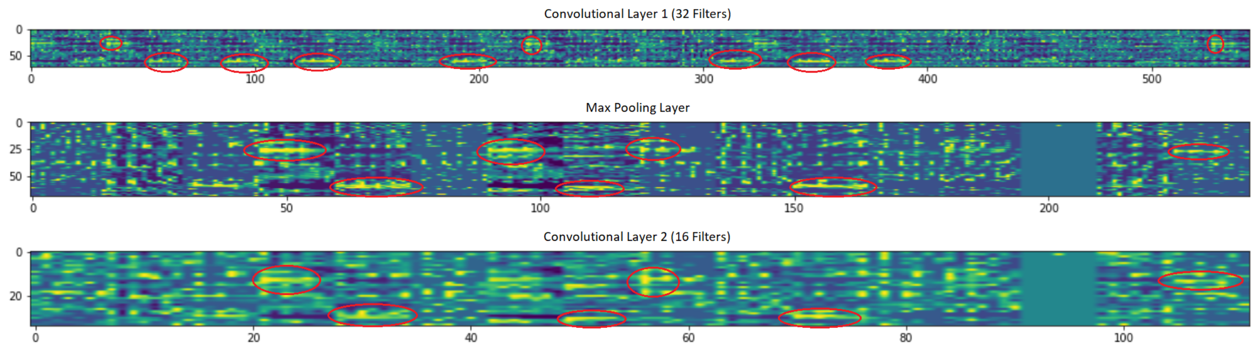


Figure 5: Feature maps of the best CNN model for a one-month-ahead ($h = 1$) forecast horizon. The plots represents the values provided to the data points by the filters in the first convolutional layer, the first max-pooling layer and the second convolutional layers. For the first convolutional layer, the plot contains 32 filters with each 18 (half of the 36 months used as input to the model) values on the x-axis and 77 (half of the 144 input variables) values on y-axis. As such, a new filter starts each 18 horizontal steps. These values are summarized in a max-pooling layer. The second convolutional layer contains 16 filters with 8 values each on the x-axis and 38 values on the y-axis. The brighter the data point is, the more value is attributed to this data point by the filter and subsequently the CNN. Important areas within the plots are highlighted with red circles.

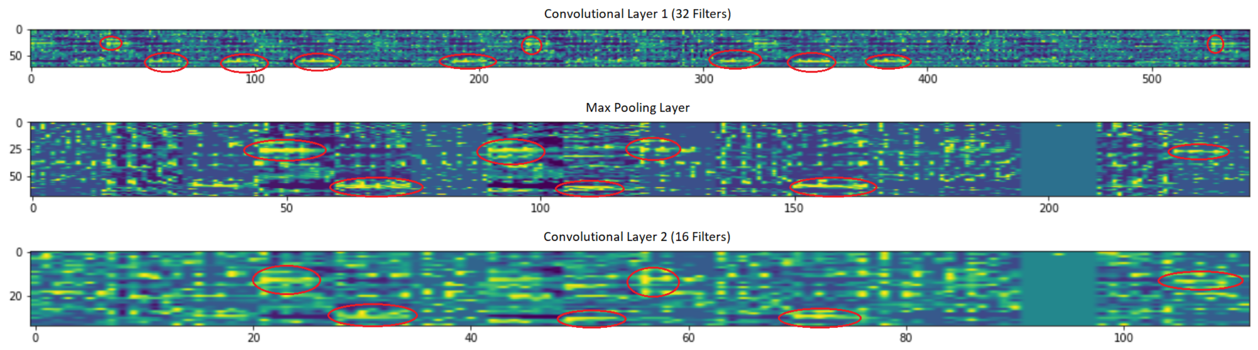


Figure 6: Feature maps of the best CNN model for a three-month-ahead ($h = 3$) forecast horizon. The plots represents the values provided to the data points by the filters in the first convolutional layer, the first max-pooling layer and the second convolutional layers. For the first convolutional layer, the plot contains 8 filters with each 18 (half of the 36 months used as input to the model) values on the x-axis and 77 (half of the 144 input variables) values on y-axis. As such, a new filter starts each 18 horizontal steps. These values are summarized in a max-pooling layer. The second convolutional layer contains 64 filters with 8 values each on the x-axis and 38 values on the y-axis. The brighter the data point is, the more value is attributed to this data point by the filter and subsequently the CNN. Important areas within the plots are highlighted with red circles.

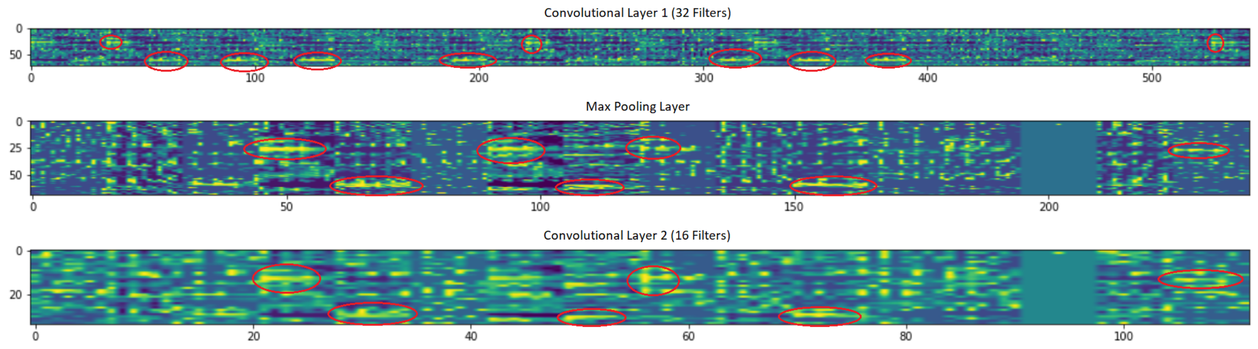


Figure 7: Feature maps of the best CNN model for a twelve-month-ahead ($h = 12$) forecast horizon. The plots represents the values provided to the data points by the filters in the first convolutional layer, the first max-pooling layer and the second convolutional layers. For the first convolutional layer, the plot contains 32 filters with each 18 (half of the 36 months used as input to the model) values on the x-axis and 77 (half of the 144 input variables) values on y-axis. As such, a new filter starts each 18 horizontal steps. These values are summarized in a max-pooling layer. The second convolutional layer contains 64 filters with 8 values each on the x-axis and 38 values on the y-axis. The brighter the data point is, the more value is attributed to this data point by the filter and subsequently the CNN. Important areas within the plots are highlighted with red circles.

Appendix E. Code Description

This Appendix contains a description of the Python files used for implementing the different methods used in this thesis. Additional description regarding the code can be found in the individual files. To prevent a very long appendix with code, I would like to refer to the file "Code_Bachelor_Thesis_Sean_Goedgeluk_443106.zip", which contains the following Python files:

dataprocess.py

This file contains the necessary functions to import the data set containing the monthly macroeconomic data of the United States, as well as import that transformation codes by Kim & Swanson (2018). On top of this, it contains the function to preprocess the data using the transformation codes and save this preprocessed data.

boosting.py

This file implements the boosting and principal component analysis (PCA) method. The code and methods used by Kim & Swanson (2018) were used as inspiration for this.

models.py

This file covers the autoregressive (AR) model by Kim & Swanson (2018) and the AR model that uses both OLS and maximum likelihood to estimate the model. On top of this, it includes the Diebold-Mariano test and functions to provide a summary and comparison of the accuracy of the model(s).

deeplearning.py

This files contains the implementation of the three deep learning models used in this paper: the long short-term memory (LSTM) neural network, the convolutional neural network (CNN) and the CNN-LSTM combination. These models are split up into functions for the empirical study, or estimation of the neural network based on set parameters, and those for the simulation study, which use Hyperband optimization to find the best possible parameters. Additionally, it contains scaling and descaling functions for the data, as this is required for proper functioning of the neural networks. Lastly, it also includes a function to create the input for the neural networks, as this requires several processing steps.

USreplication.py

This file is used to combine the boosting and AR methods and replicate the study performed by Kim & Swanson (2018). It does not apply any deep learning models to the data.

USnew.py

While the previous file looks at the replication of the study by Kim & Swanson (2018). This file implements the new study performed in this paper. It implements the different models for the new data setting and generates the forecasts for all models, saving these to the computer. It also contains a function to plot the feature maps seen in Appendix D.

simulation.py

This file contains the code to create the simulation data sets and create the forecasts for all the different models. It also performs the comparison of the forecast accuracy.