engineering proceedings

MDPI

Proceeding Paper

A CNN–BiLSTM Architecture for Macroeconomic Time Series Forecasting †

Alessio Staffini

Department of Economics and Finance, Catholic University of Milan, 20123 Milan, Italy; alessio.staffini@bocconialumni.it

† Presented at the 9th International Conference on Time Series and Forecasting, Gran Canaria, Spain, 12–14 July 2023.

Abstract: In forecasting socio-economic processes, it is essential to have tools that are highly performing, with results as close to reality as possible. Forecasting plays an important role in shaping the decisions of governments and central banks about macroeconomic planning, and it is an essential analytical tool in defining economic strategies of countries. The most common forecasting methods used in the analysis of macroeconomic processes are based on extrapolation, i.e., extending the trend observed in the past (and present) to the future. However, the presence of non-linearity in the socio-economic systems under uncertainty, as well as the partial observability of the processes, has contributed to make researchers and practitioners consider other methodologies, too. In this paper, we analyze 18 time series of macroeconomic variables of the United States of America. We compare the benchmark results obtained with "classic" forecasting techniques with those obtained with our proposed architecture. The model we construct can be defined as "hybrid" since it combines a Convolutional Neural Network (CNN) with a Bidirectional Long Short-Term Memory Network (BiLSTM) backend. We show that, for what concerns minimizing the forecast error, our model competes with and often improves the results obtained with the benchmark techniques. The goal of this work is to highlight that, due to the recent advances in computing power, new techniques can be added to the set of tools available to a policymaker for forecasting macroeconomic data.

Keywords: time series forecasting; economic forecasting; macroeconometric forecasting; deep learning; CNN-BiLSTM

Citation: Staffini, A. A CNN–BiLSTM Architecture for Macroeconomic Time Series Forecasting. *Eng. Proc.* **2023**, *39*, 33. https://doi.org/10.3390/engproc2023039033

Academic Editors: Ignacio Rojas, Hector Pomares, Luis Javier Herrera, Fernando Rojas and Olga Valenzuela

Published: 30 June 2023

1. Introduction

Machine Learning is now deeply entwined with many aspects of modern society, from optimizing web searches and filtering content on social networks to personalized recommendations on e-commerce sites. Increasingly, these applications make use of a class of techniques called Deep Learning, which forms a subfield of Machine Learning. Deep Learning techniques are based on Artificial Neural Networks (ANNs), which are vaguely inspired by the structure and functioning of the brain. ANNs are made by artificial neurons, and these neurons are organized in many different layers: a given layer computes the values to be used as input for the next layer in order to process the data in an increasingly complex and complete way.

With a sufficient amount of data, such networks are able to learn the correct representation and to outperform Machine Learning algorithms [1,2]; in other words, Deep Learning can be thought of as a learning technique in which ANNs are exposed to very huge amounts of data so that they can efficiently learn to perform tasks [3].

Deep Learning is one of the main sources of success for the recent advances in Artificial Intelligence: because of ANNs and their variations, which can efficiently analyze images, video, audio and sequential data, this area is developing rapidly.

For some years now, researchers have been trying to apply Deep Learning for forecasting analysis on time series data. Excellent results have been achieved in some fields (among

all, that of weather forecasts [4,5]), but in the economic–financial field, the use of these new tools has not yet matured except for some applications in portfolio selection [6,7] and in stock price forecasting [8–10]. Even so, Machine Learning techniques and Deep Learning algorithms have shed light on new approaches to dealing with prediction problems, in which the relationships between input and output variables are modeled in a deep and layered hierarchy. Machine Learning techniques such as Support Vector Machines (SVMs) and Random Forests, as well as Deep Learning techniques such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) have gained a lot of attention recently, and they are increasingly being applied in many disciplines [11–13]. Unlike simpler classification and regression problems, time series forecasting adds a further order of complexity, caused by the time dependence between observations. Traditionally, time series forecasting has been dominated by linear methods, such as ARIMA, because they have a good theoretical framework and are effective on many problems [11,14]. However, these methods also suffer from some limitations, such as being sensible to the presence of missing data, excluding complex non-linear mappings between inputs and outputs, and requiring a careful data preparation to find the optimal number of delays.

Forecasting time series data in the macroeconomic domain is an even more challenging task, mainly due to the possible unprecedent changes in economic trends on the one hand, and incomplete information on the other [15–17]. The market volatility in recent years has created serious concerns over the precision of economic and financial forecasts; therefore, the accuracy of the predictions is an extremely important parameter to consider when selecting a forecasting method [18,19].

One of the goals of this paper is to study which techniques offer the best forecasts, starting from a dataset of US macroeconomic data. In this regard, we first employed three econometric models (ARIMA, ARIMAX, and VAR), as well as Machine Learning models (Elastic Net, XGBoost, and BiLSTM), to measure their prediction accuracy on the examined variables. We have chosen to use these models because they are the most popular forecasting tools in the econometric field [20,21] or because they have been deemed as optimal for forecasting certain macroeconomic variables [15] and because, in a comparison with other techniques (naive forecasting, exponential smoothing, etc.) they demonstrated better performances. Next, we measured the prediction accuracy reported by our Deep Learning architecture.

Deep Learning methods add non-linearity and complexity in time series forecasting, and therefore they can provide good performance [3]. Furthermore, neural networks should be able to exploit all the capabilities of classical linear prediction methods, since they basically try to find the best mapping from inputs to outputs [2]. However, when it comes to time series forecasting, ANNs and complex methods should be evaluated and applied carefully, as depending on the data at hand they do not always deliver better results than more traditional tools [22,23]. It is therefore important to include and evaluate the performance of multiple methods to justify the application of neural networks.

CNNs are feedforward neural networks with shared weights and sparse interactions, often used for processing data that has a known grid-like topology (such as images or time series data). In CNN architectures, it is custom to process the data with a certain number of convolutional layers, followed by a max-pooling layer. Informally, the convolutional layers filter the data by removing noise and extracting features of the input, while the max-pooling layer acts as an information contractor, keeping only a summary of those filtered features. RNNs are a family of neural networks obtained by adding feedback connections to the feedforward architecture, and they are again used mainly for processing sequential data (such as text and time series data); LSTMs are a particular kind of them, which introduced the idea of gates to better control the flow of information and learn long-term dependencies.

The main goal of this paper is to show that, due to the recent advances in computing power, there are additional methods of analysis that are available to policymakers,

especially when they are planning some interventions based on the forecasts of economic variables. We suggest that these methods should be taken into consideration.

2. The Model

Our proposed architecture is a hybrid CNN-BiLSTM. First, we stacked two one-dimensional convolutional layers (64 filters each, with a filter size of 3) to filter the input signal. The results of these convolutions were then passed to a one-dimensional MaxPooling layer (pooling size of 2, no strides and "valid" padding), in order to down-sample the feature maps, reduce the dimensionality, extract only the relevant information and in general make the network more robust. Notice that we only halved the dimensionality, applying a modest pooling size of 2. Applying a pooling layer after (usually two or more) convolutional layers in order to filter the relevant information while also making the network easier to train is a standard procedure in CNNs, even though recently (see, for example, [24]) some researchers proposed efficient architectures made only of convolutional layers that often reached state-of-the-art performance in a series of object recognition tasks. We then flattened the down-sampled results into a vector of neurons, and we provided it as a sequence to three-layer stacked Bidirectional LSTMs (BiLSTMs), each one, respectively, with 200, 100 and 50 units. A BiLSTM cell [25] can be simply thought of as made of two LSTMs: one reading the input sequence forward and the other reading it backwards, before concatenating both interpretations. Doing so increases the amount of information available to the model because it provides more context and, while so far it has been typically applied for Natural Language Processing tasks, it recently proved to be very useful also for time series forecasting [13].

Finally, the output of the BiLSTM submodel is followed by a fully connected (FC) layer of 25 neurons to interpret its outcomes. We then stacked the output layer, which was again a fully connected one, this time made by 12 neurons (when we wanted to forecast the next 12 months) or just a single neuron (when we performed single-step forecasting).

When training complex Deep Learning models, overfitting the train set is a very common phenomenon. To deal with this problem and provide a good generalization, we applied extensive dropout. Dropout [26] is an algorithm for which, at every training step, each neuron in the considered layer has a certain defined probability p of being "dropped". Dropout offers regularization because it approximates training in parallel many neural networks with different architectures: since at each training step a neuron can be active or not, we are effectively training 2^N different networks (where N represents the number of neurons to which dropout is applied). It is also a very suitable technique when we do not have a high amount of training data [27,28].

We applied dropout after the CNN submodel, after each BiLSTM layer, and after the FC layer (before the output layer), for a total of five dropout layers. We applied stronger dropout ($p = 0.5$) after each BiLSTM layer, while after the CNN submodel and the FC layer we applied a milder one ($p = 0.1$).

As highlighted by [26], dropout works even better when combined with other regularization techniques, such as weight and bias constraints. Large weight size can lead to the problem of exploding gradients [29] which in turn makes the network unstable. To deal with this problem, we can force the norm of all the weights in a layer to be below a certain threshold c. Following again [26], we combined dropout with max-norm regularization, bounding both the weights and the biases of each hidden layer to have their norms be below three.

The activation function of a neuron/node is used to map the weighted sum of the inputs to the output. For the entire network, we applied a ReLU (Rectified Linear Unit) activation function, which is a piecewise linear function that outputs the summed weighted input if positive, otherwise it outputs zero:

$$ReLU(x) = \max(0, x). \quad (1)$$

Non-linearity is needed in activation functions to learn complex enough (non-linear) mappings from the inputs to the output. ReLU and its variants have become the standard choice as activation functions for training deep neural networks because of their many desirable properties, one among which is the fact that they deal well with the problem of vanishing gradients [30]. Traditionally, for the LSTM cells, the hyperbolic tangent (*tanh*) function is used as activation. However, empirically, we found that passing the BiLSTM output as argument to a ReLU function performed slightly better, which is in line with other results in the literature [31]. We initialized the weights of each layer applying the Glorot uniform initializer [30], which in practice works well for deep networks [32].

Another important choice is the optimization algorithm. We selected Adam [33] because it combines both the ideas of momentum estimation and RMSProp, and it is fast and efficient. It is typically the default choice [34], and indeed in our experiments we found it to perform the best.

Finally, we trained our network using mini-batch gradient descent, which works as a compromise between gradient descent and stochastic gradient descent (SGD). Using mini-batches, we obtained a more stable convergence than using SGD while still injecting enough noise in the model, which is required for non-convex optimization [35]. We chose mini-batches of size $m = 32$, which usually works well in practice [36,37].

Figure 1 shows a stylized picture of our architecture.

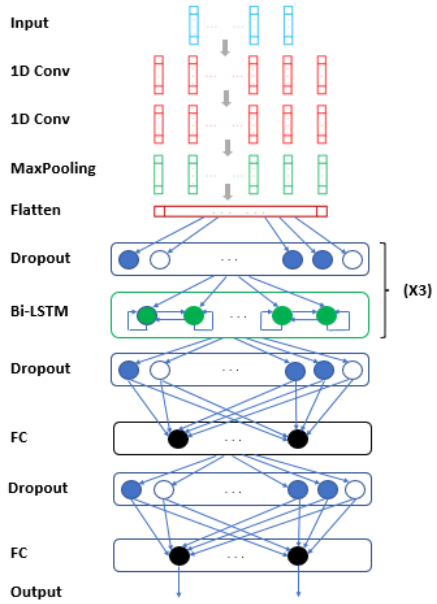

Figure 1. The network structure of our CNN-BiLSTM architecture.

Consider a normalized time series X_n, where n denotes the number of input variables ($n = 1$ for the univariate case) and $y \in X_n$ represents our variable of interest. After reframing the data as a supervised learning problem (where a given segment of X_n is used to forecast the next t steps of y—which we denote here as y_t), our described model first processes X_n by extracting different features with different filters, down-sampling the resulting feature maps via MaxPooling. The resulting outcome F is then flattened and provided as input to the BiLSTM backend, which further processes the data and makes the final prediction, $\hat{y}_t$. In short,

$$F = \text{CNN}(X_n), \tag{2}$$

$$\hat{y}_t = \text{BiLSTM}(F). \tag{3}$$

3. Empirical Analysis

We conducted our empirical analysis using data collected by the Bureau of Economic Analysis (BEA) and the Bureau of Labor Statistics (BLS) on different economic indicators of the United States of America. Among the many variables collected by the two agencies, we chose those that presented a low correlation (see Figure 2) in order to capture different aspects of the economy as a whole. The 18 chosen variables have already been seasonally adjusted, and they range from June 1947 to December 2019. They were measured monthly (we linearly interpolated those that were recorded on a quarterly basis).

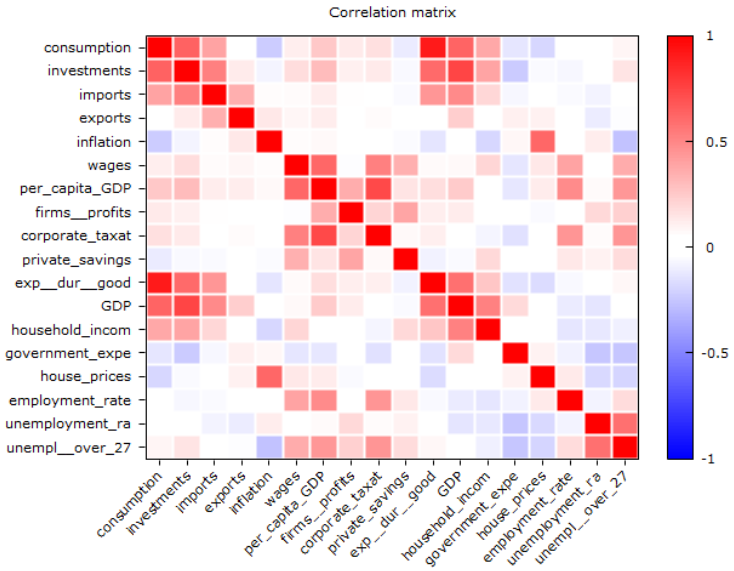

Figure 2. Correlation matrix of the considered variables.

The collected variables are consumption of goods, expenditure on durable goods, private investments, import, export, inflation, GDP, available household income, government expenditure, house prices, employment rate, and unemployment rate (all expressed as growth rate over the same period of the previous year); average wages, Per Capita GDP (in dollars); corporate profits, corporate taxation, private savings (in billions of dollars); long-term unemployment (as a percentage of total unemployment).

Each variable is numerical, and we had no missing values in the original dataset. We had a total of 871 observations for each variable. We split each time series into a train set and a test set, using an 80–20 split ratio (i.e., the first 697 values were used for training our models while the last 174 values were used to evaluate their performance). The reason behind this split is because it is one of the of the most common ones and has been shown to work well in practice [38]. We ran both multi-step forecasting (where the models tried to forecast at once the next 12 values) and one-step forecasting (where the models iteratively forecasted the next time step).

We first implemented three "classic" econometric models to perform forecasting on the test set: an ARIMA model, an ARIMAX model, and a VAR model. We also considered three Machine Learning models: a regularized linear regression model (Elastic Net), a gradient boosting algorithm (XGBoost), and a baseline Deep Learning architecture (BiLSTM). The forecast errors obtained from these models were the benchmark results to which to compare our proposed architecture.

First, we needed to verify that the considered processes were stationary. We applied the Augmented Dickey Fuller (ADF) test to check for the existence of unit roots. For only 5 of the 18 variables (average wages, Per Capita GDP, corporate profits, corporate taxation, and private savings) we did not reject the null hypothesis of having a unit root at a significance level of 0.01. We set for these variables the differencing term to $d = 1$ in the ARIMA and ARIMAX models, and a further test on the differentiated data of these variables confirmed

the absence of unit roots. In addition, the autocorrelation function (ACF) and the partial autocorrelation function (PACF) of these five variables further suggested the idea that $d = 1$. We employed ACF and PACF also for finding the appropriate order of the autoregressive term p and the moving average term q. We further compared the model specification suggested by ACF and PACF with the one suggested by information criteria, such as the Akaike Information Criterion (AIC). The specification suggested by ACF/PACF and AIC often matched; when it did not, we preferred the one that empirically performed better. This selection criterion led to the definition of different models depending on the variable. In regard to the specifications for ARIMAX and VAR models, we selected the additional covariates relying on the results of the Granger causality test, i.e., we included only further explanatory variables that "Granger-cause" the dependent one; this can be understood as including only the variables that improve the forecast results of the dependent one.

The choices of the values of p and q in the ARIMAX models were conducted as for the ARIMA case. Naturally, since we were also considering further covariates, the implemented specifications were different from those of ARIMA.

The number of lags p in the VAR models were selected using the multivariate version of AIC. Since for each VAR model we included different covariates in the system (recall that we selected only those that passed the Granger causality test for the considered dependent variable of interest), we obtained different specifications for each model. We also considered the possible presence of cointegration but, given the few processes that were $I(1)$ in first place, we never encountered such a case. We ran both the Engle–Granger cointegration test and the Johansen cointegration test, and indeed they excluded such a possibility. Therefore, when we had one or more $I(1)$ processes in the system, we implemented a VAR in differences, without the risk of it being mis-specified.

For what concerns Elastic Net and XGBoost, we used a grid search approach on the train set of each variable to optimize the values of the hyperparameters as well as the number of inputs, testing the performance of multiple different configurations and choosing the most performing one. The BiLSTM architecture is the same three-layer stacked configuration chosen for the BiLSTM backend in our proposed CNN-BiLSTM model.

4. Results

After having implemented the econometric models, we ran diagnostic checks to verify the homoskedasticity, normality and absence of autocorrelation in the residuals: all the models passed these tests, confirming the correctness of the selected specifications.

In order to compare the performance of the results obtained with the baseline models with that obtained from our CNN-BiLSTM architecture, we needed to select an error metric. We chose two popular metrics regularly employed in evaluation studies: the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE), which is the square root of MSE. MAE is considered a natural measure of average error [39], while RMSE has the property of penalizing large errors more, and it is often the preferred measure for regression tasks in absence of many outliers. Considering N observations, their equations are given by

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{4}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{5}$$

where y_i is the true value and $\hat{y}_i$ is the value predicted by the model.

The econometric models have been implemented using the statistical package gretl (Version 2020b), while the Machine Learning models and our proposed architecture were built with Python (Version 3.7.3). We collected all the results in Tables 1–4.

Table 1. Multi-step forecast errors for the univariate input models. Text in bold denotes the best performance (95% confidence level).

Variable	ARIMA	Elastic Net	XGBoost	BiLSTM	Our CNN-BiLSTM
	RMSE MAE	RMSE MAE	RMSE MAE	RMSE (std. dev.) MAE (std. dev.)	RMSE (std. dev.) MAE (std. dev.)
consumption	4.0965 2.8597	2.9343 1.9814	3.0921 2.0763	2.9725 (0.0686) 2.0737 (0.0217)	**1.9736 (0.0463)** **1.5941 (0.0459)**
investments	8.8201 6.0830	6.0735 4.0524	6.1441 3.9671	6.2068 (0.7248) 4.1441 (0.4170)	**3.7446 (0.1710)** **2.9190 (0.1372)**
imports	11.088 7.1216	8.3575 6.1214	7.7900 5.2379	7.8201 (0.4894) 5.1850 (0.4264)	**4.3390 (0.1758)** **3.3621 (0.1416)**
exports	10.599 6.6301	7.1607 5.0656	7.2648 4.9356	7.1996 (0.2561) 4.6305 (0.4166)	**5.1139 (0.1191)** **3.3972 (0.0941)**
inflation	2.2526 1.4850	1.5051 0.9833	1.5849 0.9830	1.5760 (0.0418) 1.0704 (0.0357)	**0.8927 (0.0305)** **0.6801 (0.0404)**
wages	206.60 143.02	148.31 101.07	166.21 127.92	164.75 (8.7017) 130.71 (13.638)	**141.20 (2.4414)** **95.986 (3.4415)**
per capita GDP	145.61 107.06	102.61 76.707	114.19 87.994	148.54 (9.9937) 125.79 (11.249)	**86.209 (2.9799)** **68.618 (3.3470)**
firms' profits	39.849 28.648	28.078 19.137	29.166 19.690	29.548 (1.0818) 19.660 (0.7318)	**23.911 (0.9131)** **18.481 (0.5890)**
corporate taxation	4.0462 3.1330	3.2166 2.4978	3.7096 2.9433	4.5836 (0.6401) 3.6909 (0.2091)	**3.0049 (0.0761)** **2.3460 (0.0533)**
private savings	47.670 35.448	37.938 27.982	38.543 28.764	39.569 (1.5244) 29.538 (1.2444)	**36.840 (03155)** **27.392 (0.3602)**
expen. durable goods	8.5890 5.9085	6.0217 3.8684	6.3203 4.2891	6.5361 (0.1523) 4.4667 (0.1151)	**4.2486 (0.0985)** **3.2753 (0.0737)**
GDP	2.9405 2.0696	2.2276 1.6256	2.2654 1.5530	2.1998 (0.1925) 1.4258 (0.0305)	**1.3007 (0.0669)** **0.9965 (0.0445)**
household income	4.4913 2.9752	3.0415 2.0485	3.3690 2.1789	3.0825 (0.0092) 1.9547 (0.0143)	**2.9496 (0.0172)** **1.8623 (0.0161)**
government expenditure	2.4130 1.9668	2.4078 1.8971	2.6883 2.0271	2.4379 (0.0322) 2.0667 (0.0413)	**1.9379 (0.0894)** **1.5853 (0.0892)**
house prices	5.5399 4.2488	2.8767 1.9062	3.4029 2.1275	3.6669 (0.1034) 2.5121 (0.1170)	**2.3246 (0.0597)** **1.8021 (0.0506)**
employment rate	0.9396 0.5943	**0.4462** **0.2688**	0.6097 0.3803	1.0915 (0.1840) 0.6800 (0.2310)	0.8562 (0.0315) 0.7342 (0.0032)
unemployment rate	1.2203 0.8644	0.5638 0.4000	0.7087 0.4847	0.6782 (0.0911) 0.4163 (0.0714)	**0.4307 (0.0561)** **0.3565 (0.0417)**
unemp. over 27 weeks	5.8469 3.8736	3.9508 2.9983	11.856 8.7233	5.4433 (0.8422) 4.0504 (0.6085)	**1.9146 (0.1641)** **1.4947 (0.1287)**

Table 2. Single-step forecast errors for the univariate input models. Text in bold denotes the best performance (95% confidence level).

Variable	ARIMA	Elastic Net	XGBoost	BiLSTM	Our CNN-BiLSTM
	RMSE MAE	RMSE MAE	RMSE MAE	RMSE (std. dev.) MAE (std. dev.)	RMSE (std. dev.) MAE (std. dev.)
consumption	0.9643 0.7120	0.9553 0.6583	1.3197 0.9289	0.9081 (0.0280) 0.6078 (0.0211)	**0.8653 (0.0238)** **0.5385 (0.0168)**
investments	1.4326 0.9711	1.4369 0.9419	2.3368 1.4167	1.4349 (0.0392) 0.9849 (0.0383)	**1.3400 (0.0365)** **0.9246 (0.0147)**
imports	1.7607 1.1129	2.3095 1.6583	2.5519 1.7599	1.7439 (0.1267) 1.1268 (0.0544)	**1.5588 (0.0478)** **1.0276 (0.0273)**
exports	2.283 1.6165	2.2169 1.5940	2.7194 2.0555	2.3930 (0.1076) 1.5815 (0.0847)	**2.0841 (0.0452)** **1.3116 (0.0749)**
inflation	**0.4846** **0.2919**	0.9553 0.6593	0.6456 0.4136	0.6115 (0.0376) 0.3695 (0.0223)	0.5554 (0.0149) 0.3225 (0.0204)
wages	117.50 50.580	**98.920** **44.557**	127.54 78.866	126.99 (4.5553) 81.072 (9.1984)	121.86 (1.1490) 71.469 (2.7451)
per capita GDP	65.879 37.380	65.311 **32.470**	78.466 46.829	80.204 (5.5873) 56.759 (6.7406)	**64.167 (0.4711)** 40.493 (1.7013)
firms' profits	20.872 10.754	19.399 9.1343	23.328 13.805	20.771 (1.5599) 12.611 (0.8903)	**17.165 (0.0945)** **9.4561 (0.1147)**
corporate taxation	2.2362 1.3484	2.2831 1.2905	2.3848 1.4029	2.5278 (0.1010) 1.7611 (0.1016)	**2.0766 (0.0557)** **1.1868 (0.0732)**
private savings	26.963 18.998	28.001 20.012	34.706 23.230	31.628 (2.3708) 21.659 (1.8836)	**25.765 (0.3857)** **15.805 (08.280)**
expen. durable goods	2.1026 1.3557	2.0493 1.4233	3.2429 2.1295	1.9780 (0.1184) 1.4542 (0.0796)	**1.7694 (0.0524)** **1.2527 (0.0757)**
GDP	0.6006 0.4002	0.6097 0.3924	0.7673 0.5539	0.5439 (0.0094) 0.3475 (0.0222)	**0.5086 (0.0291)** **0.3165 (0.0240)**
household income	0.9904 0.5434	1.1643 0.6927	1.7724 0.9226	1.1304 (0.0906) 0.6783 (0.0442)	**0.8914 (0.0281)** 0.6085 (0.0277)
government expenditure	0.5634 0.4001	0.6093 0.4405	0.7855 0.6336	0.5975 (0.0394) 0.4525 (0.0304)	**0.5191 (0.0098)** **0.3923 (0.0062)**
house prices	0.7943 0.5159	**0.7671** **0.4363**	1.2116 0.7466	0.9762 (0.0797) 0.6241 (0.0592)	0.8293 (0.0291) 0.5085 (0.0313)
employment rate	**0.1316** **0.1029**	0.1373 0.1059	0.2263 0.1703	0.3539 (0.0297) 0.2360 (0.0404)	0.3716 (0.0058) 0.2422 (0.0141)
unemployment rate	**0.1547** **0.1238**	0.1592 0.1279	0.2037 0.1526	**0.1552 (0.0022)** **0.1230 (0.0027)**	0.1569 (0.0007) 0.1247 (0.0011)
unemp. over 27 weeks	**1.2845** **0.9715**	1.2896 0.9756	9.7659 6.6189	1.3294 (0.0988) 1.0333 (0.0757)	1.5099 (0.0656) 1.1661 (0.0587)

Table 3. Multi-step forecast errors for the multivariate input models. Text in bold denotes the best performance (95% confidence level).

Variable	ARIMAX	VAR	Elastic Net	XGBoost	BiLSTM	Our CNN-BiLSTM
	RMSE MAE	RMSE MAE	RMSE MAE	RMSE MAE	RMSE (std. dev.) MAE (std. dev.)	RMSE (std. dev.) MAE (std. dev.)
consumption	4.0460 2.7761	4.1935 3.0393	3.6708 2.6287	3.4773 2.3817	3.4177 (0.1048) 2.5099 (0.0734)	**3.2332 (0.0232)** **2.2925 (0.0376)**
investments	8.3121 5.4483	11.113 8.8338	6.9493 4.5236	6.4072 4.2031	6.2005 (0.3294) 4.2836 (0.2893)	**5.6495 (0.0631)** **3.8215 (0.1032)**
imports	9.5076 7.1720	8.7729 6.8316	7.6435 5.4115	7.8722 5.5894	6.9976 (0.1286) 4.3225 (0.1295)	**6.5982 (0.0757)** **4.0813 (0.0615)**
exports	8.6620 6.4796	8.7729 6.8316	7.6435 5.4115	7.3792 5.3136	7.3524 (0.1209) 5.2083 (0.2321)	**6.9591 (0.1499)** **4.7479 (0.1894)**
inflation	1.9595 1.5155	1.7622 1.1454	1.6580 1.2344	**1.5018** **0.9855**	**1.5093 (0.0164)** 1.0497 (0.0339)	**1.5165 (0.0353)** 1.0930 (0.0473)
wages	173.51 135.74	164.74 123.20	142.60 94.343	147.79 99.890	154.29 (7.3877) 108.36 (11.398)	**141.65 (0.4800)** **91.314 (1.1180)**
per capita GDP	131.39 103.27	127.81 99.720	108.66 80.566	109.46 84.470	104.87 (2.6097) 79.508 (2.7268)	**97.836 (0.5310)** **72.670 (0.4794)**
firms' profits	29.515 20.162	29.606 20.127	29.169 20.061	28.808 19.758	28.881 (0.1190) 19.517 (0.1231)	**28.029 (0.0803)** **18.927 (0.1183)**
corporate taxation	4.1696 3.0078	3.9330 2.8608	3.0939 2.3600	3.5095 2.7044	3.2635 (0.0528) 2.5187 (0.0696)	**2.9983 (0.0300)** **2.2922 (0.0125)**
private savings	39.156 29.315	40.942 30.340	37.644 27.215	37.657 27.197	36.509 (0.3576) 26.957 (0.1213)	**35.862 (0.2169)** **26.416 (0.0249)**
expen. durable goods	7.5644 5.0644	6.9866 5.3048	6.7296 4.3630	6.6235 4.4179	6.7366 (0.1282) 4.7585 (0.2162)	**6.3273 (0.1463)** **4.2023 (0.0660)**
GDP	3.2227 2.4332	3.3951 2.3968	2.5851 1.8965	2.3209 1.6542	2.2353 (0.1251) 1.6550 (0.2241)	**2.0479 (0.0587)** **1.3924 (0.0620)**
household income	3.5447 2.5716	3.0540 2.0809	3.1000 2.1340	3.4116 2.1781	3.0019 (0.0269) 2.0218 (0.0438)	**2.8639 (0.1174)** **1.9530 (0.0519)**
government expen.	3.5540 2.7296	4.8023 3.4904	2.0151 1.5048	2.3825 1.8464	2.3299 (0.0241) 1.8869 (0.0480)	**1.5517 (0.0455)** **1.2469 (0.0295)**
house prices	4.1156 2.7895	5.2339 4.0954	3.4730 2.5971	3.4337 2.4875	3.6274 (0.0694) 2.7029 (0.3359)	**3.2234 (0.0437)** **2.3406 (0.0359)**
employment rate	2.0070 1.5946	4.2618 3.9164	0.4970 0.3281	0.5008 0.3830	0.5733 (0.1733) 0.4789 (0.1207)	**0.3754 (0.0168)** **0.2988 (0.0182)**
unemployment rate	2.0894 1.6533	2.3794 1.7583	0.5858 0.4443	0.5554 0.4153	0.6595 (0.0508) 0.4667 (0.0342)	**0.4636 (0.0306)** **0.3147 (0.0314)**
unemp. over 27 w.	17.772 15.120	17.696 17.218	2.9366 2.2811	9.5743 6.9341	3.7248 (0.6547) 3.1515 (0.5486)	**2.8321 (0.0330)** **2.2403 (0.0075)**

Table 4. Single-step forecast errors for the multivariate input models. Text in bold denotes the best performance (95% confidence level).

Variable	ARIMAX	VAR	Elastic Net	XGBoost	BiLSTM	Our CNN-BiLSTM
	RMSE MAE	RMSE MAE	RMSE MAE	RMSE MAE	RMSE (std. dev.) MAE (std. dev.)	RMSE (std. dev.) MAE (std. dev.)
consumption	1.7566 1.2117	2.9969 2.4505	**1.1707** **0.8992**	1.6084 1.2032	1.9307 (0.1019) 1.4358 (0.1144)	2.1592 (0.1323) 1.6629 (0.1393)
investments	5.3641 3.6766	10.213 9.2144	2.7433 2.2516	2.8168 1.8298	2.5233 (0.1986) **1.8002 (0.1657)**	**2.2397 (0.2277)** **1.6682 (0.1700)**
imports	5.3600 4.3036	6.0002 4.7285	2.8470 2.1585	3.4139 2.0686	3.2036 (0.2811) 2.4632 (0.1612)	**2.3958 (0.0599)** **1.7573 (0.0653)**
exports	7.5044 5.1107	7.2302 5.4215	3.9126 2.8886	3.9077 **2.7276**	4.4484 (0.3598) 3.0564 (0.1857)	**3.6799 (0.1089)** **2.7379 (0.1062)**

Table 4. *Cont.*

Variable	ARIMAX	VAR	Elastic Net	XGBoost	BiLSTM	Our CNN-BiLSTM
inflation	2.4004 1.9318	5.8533 4.6454	**0.6923** **0.4677**	0.7326 0.4734	1.3577 (0.1675) 1.0098 (0.1148)	1.1152 (0.2807) 0.8316 (0.2151)
wages	144.23 107.37	138.73 92.062	117.72 68.958	124.43 75.590	118.26 (2.1950) 68.691 (2.4276)	**114.22 (2.0562)** **63.500 (1.9796)**
per capita GDP	87.899 62.521	78.678 59.400	64.985 45.917	83.200 54.800	71.752 (2.4801) 49.911 (0.9571)	**61.601 (1.3644)** **42.292 (2.1096)**
firms' profits	23.038 16.984	24.186 17.132	22.729 12.229	28.622 18.731	21.102 (0.6201) 13.122 (0.3800)	**18.358 (0.3645)** **11.501 (0.6107)**
corporate taxation	3.5300 2.6559	3.2948 2.6556	2.3399 1.5964	2.5744 1.5975	2.4300 (0.1239) 1.6751 (0.0506)	**2.2880 (0.0093)** **1.5749 (0.0094)**
private savings	29.915 21.136	32.496 22.814	28.248 16.479	37.090 23.940	33.859 (1.1122) 22.989 (1.0929)	**25.749 (0.4926)** **15.895 (0.2147)**
expen. durable goods	7.6315 4.9930	15.597 12.497	5.2789 3.4234	5.0983 3.2737	7.0292 (0.3995) 5.0255 (0.7138)	**4.2094 (0.1529)** **3.1987 (0.1189)**
GDP	1.3641 0.9897	2.2758 1.9223	0.9309 0.7082	0.9349 0.7232	1.1034 (0.0647) 0.7880 (0.0315)	**0.8727 (0.0372)** **0.6515 (0.0343)**
household income	3.1828 2.2418	7.3236 5.7770	2.1474 1.4936	**1.9454** **1.0294**	2.2263 (0.1348) 1.5581 (0.1717)	2.2520 (0.1713) 1.6404 (0.1257)
government expen.	3.8283 3.1686	2.8563 2.2076	1.2604 1.0073	**1.0056** **0.8085**	1.1806 (0.1617) **0.8747 (0.1080)**	1.0664 (0.0420) **0.8037 (0.0320)**
house prices	4.0064 2.7954	3.2791 2.7633	**0.7930** **0.6006**	1.3561 0.9131	1.3317 (0.1338) 1.0029 (0.1015)	1.2537 (0.0430) 0.9268 (0.0668)
employment rate	2.0802 1.6528	0.6177 0.4981	**0.1319** **0.1027**	0.2055 0.1618	0.2540 (0.0274) 0.1981 (0.0160)	0.2182 (0.0099) 0.1700 (0.0067)
unemployment rate	1.8229 1.3537	2.2742 1.8059	**0.1414** **0.1287**	0.3147 0.2085	0.2357 (0.0259) 0.1915 (0.0240)	0.2013 (0.0022) 0.1566 (0.0021)
unemp. over 27 w.	19.020 16.049	12.429 11.726	**1.1769** **0.9036**	10.796 7.5981	1.9132 (0.2058) 1.4810 (0.2187)	1.7467 (0.0925) 1.3394 (0.0701)

For the sake of equal comparisons, we compared the MAE/RMSE of the ARIMA models and univariate Elastic Net/XGBoost/BiLSTM with the MAE/RMSE obtained from a univariate input CNN-BiLSTM (that used only lagged values of the dependent variable), and we compared the MAE/RMSE of the ARIMAX, VAR, and multivariate Elastic Net/XGBoost/BiLSTM models with the ones obtained from a multivariate input CNN-BiLSTM (that used, in addition to the lagged values of the dependent variable, also lagged values from additional covariates). As mentioned in Section 3, we selected as additional covariates to be added to the multivariate input models only those variables that Granger-caused the dependent variable. We reported the selected variables in the Appendix A (Table A1).

In the above tables, for each variable, the text in bold denotes the best performance for the considered metric. From Tables 1 and 2, we can see that for the univariate time series analysis, the forecast errors of our architecture were in general much lower than the competitor models, in particular for the multi-step forecast. Indeed, when trying to forecast the next 12 values, the CNN-BiLSTM architecture significatively performed better for 17 variables out of 18. For the one-step forecast, there were only six variables where the results of the competitor models were significantly better, and even in those cases the results of our architecture were often close.

We obtained the same excellent results for multi-step forecasting when we operated in a multivariate time series context (see Table 3). The multivariate one-step forecast

results (Table 4) were also satisfactory: our model performed better than the multivariate competitor models for 10 variables out of 18.

It should be noted that, for all the models, the multivariate results were in general significantly worse than the univariate case. In particular, for what concerns our proposed model, we guess that one reason behind this worsening is related to the higher amount of information (coming from multiple time series) that was processed by the network: to determine the best configuration of the parameters, the network tries to minimize a non-convex cost function that lies in a very high dimensional space, and non-relevant information could make the optimization process to become more easily stuck in poor local minima. We think that a longer training combined with injecting more noise (for instance, by reducing the mini-batch size) could solve the problem, and we plan to study this issue in greater detail in future studies.

It should also be mentioned that we cannot evaluate the quality of a Deep Learning architecture from a single run. Indeed, as a consequence of the random weight initialization and the optimization process, there is an inherent stochastic component in the obtained results. The results reported in Tables 1–4 are the average mean of 10 different evaluations of the Deep Learning models, where in brackets we reported the standard deviation. For each variable, to confirm statistically significant differences in the results, we ran independent t-tests with $p < 0.05$.

We can also visualize the predictions of the competitor models with those of our architecture. As an example, let us consider the variable "Investments". We compared the one-step forecasts (Figure 3) and the multi-step forecasts (Figure 4) obtained with ARIMA, Elastic Net, and the CNN-BiLSTM architecture.

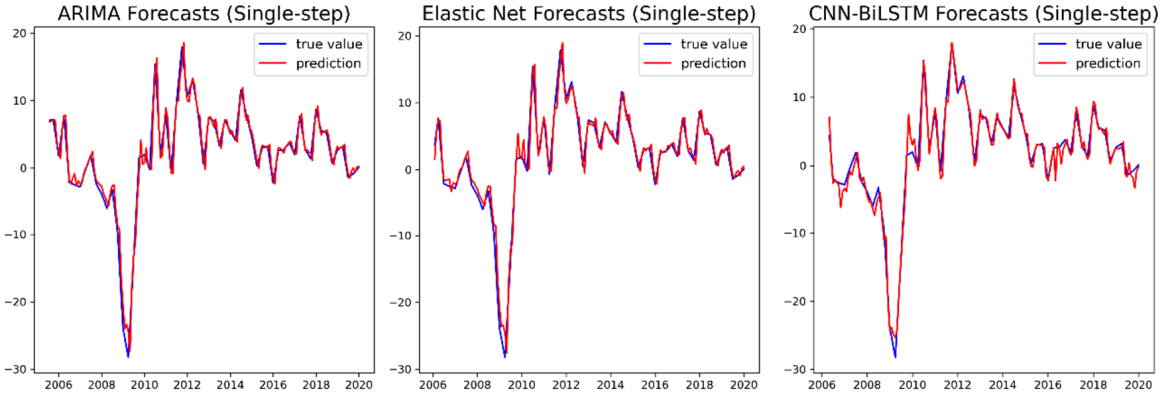

Figure 3. A comparison between the one-step forecast results for ARIMA (**left**), Elastic Net (**middle**), and our CNN-BiLSTM architecture (**right**) for the variable "Investments".

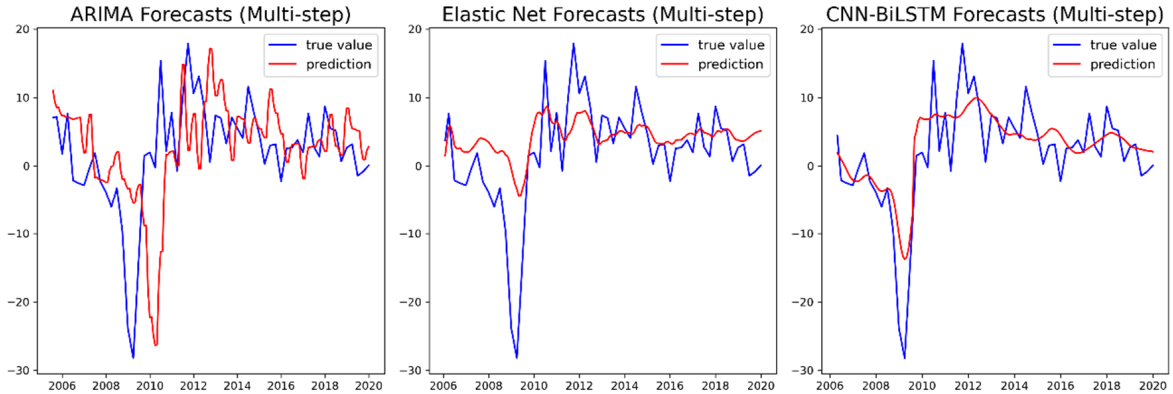

Figure 4. A comparison between the multi-step forecast results for ARIMA (**left**), Elastic Net (**middle**), and our CNN-BiLSTM architecture (**right**) for the variable "Investments".

5. Conclusions

This paper addresses the problem of obtaining increasingly accurate forecasts for different macroeconomic indicators in order to offer a valid support for policymakers to better plan economic policies. With the recent advances in Machine Learning (and in particular Deep Learning), new investigation techniques are emerging, and they are gaining popularity among researchers from different disciplines. However, in the context of macroeconomic forecasting, these techniques are not yet widely used (except for some financial applications), and other more "traditional" econometric models are often preferred.

The question we asked ourselves was to understand how much a new model implemented with these techniques was able to compete with the traditional tools; to this purpose, we selected two error metrics (MAE and RMSE) and we compared the forecast errors of the considered models. In constructing our architecture, we implemented a "hybrid" model, combining a CNN to extract the salient features of the time series and a Bidirectional LSTM backend to learn the timing relationships and perform the forecasts. While econometric models are certainly easier to fit and are more interpretable, for what purely concerns the forecasting component the results obtained with the proposed Deep Learning architecture are very promising.

The main problem when applying these techniques to time series (in particular macroeconomic time series) is that to obtain good forecast results, one in general needs a huge number of observations. Deep Learning is not suitable in contexts where the time series are too short, or where the measurements are not frequent enough. However, we can conclude that, with the data availability that increases more and more over time, Deep Learning techniques should be taken into serious consideration to make predictions about the evolution of the economy.

Our model can be further improved: first, we selected the model hyperparameters with a "trial and error" approach, and an accurate grid search is likely to lead to better results. However, the focus of this work was on producing better (or on par) results than "traditional" methods, and not on finding the best possible architecture configuration. Effective Deep Learning applications to time series analysis are being developed only recently and there is certainly room for further improvement. We believe that new more performing algorithms and architectures will be available shortly.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used for the empirical analyses have been provided by the U.S. Bureau of Economic Analysis (BEA) and the U.S. Bureau of Labor Statistics (BLS) and are publicly available. The datasets used in the empirical experiments, as well as the code of our model, are available at: https://github.com/staale92/cnn-bilstm-macroeconomic-forecasting.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A

Appendix A.1

Table A1. Additional covariates added in the multivariate input models for each dependent variable.

consumption	inflation, per capita GDP, corporate taxation, private savings, expen. durable goods, household income, unemployment rate, unempl. over 27 weeks
investments	consumption, imports, exports, per capita GDP, firms' profits, expen. durable goods, GDP, house prices, employment rate, unemployment rate, unempl. over 27 weeks

Table A1. *Cont.*

imports	consumption, exports, inflation, GDP, expen. durable goods, household income, government expenditure, house prices
exports	consumption, investments, inflation, GDP, expen. durable goods, government expenditure, unemployment rate
inflation	corporate taxation, private savings, GDP, government expenditure, unempl. over 27 weeks
wages	per capita GDP, firms' profits, corporate taxation, household income
per capita GDP	investments, wages, firms' profits, corporate taxation, private savings, GDP, household income, employment rate
firms' profits	wages, per capita GDP, Private savings, GDP
corporate taxation	wages, firms' profits, private savings, unemployment rate
private savings	wages, per capita GDP, firms' profits, household income
expen. durable good	inflation, wages, firms' profits, private savings, employment rate, unemployment rate, unempl. over 27 weeks
GDP	consumption, investments, per capita GDP, firms' profits, expen. durable goods, household income, government expenditure, house prices, unemployment rate
household income	consumption, imports, inflation, wages, firms' profits, private savings, expen. durable goods, house prices, employment rate, unemployment rate, unempl. over 27 weeks
government expenditure	investments, exports, inflation, expen. durable goods, household income, unemployment rate
house prices	imports, exports, inflation, corporate taxation, expen. durable goods, GDP, government expenditure, employment rate, unemployment rate
employment rate	consumption, inflation, per capita GDP, GDP, household income, government expenditure
unemployment rate	consumption, inflation, expen. durable goods, GDP, employment rate
unempl. over 27 weeks	private savings, imports, exports, GDP, house prices, unemployment rate

Appendix A.2. Specifications of the ARIMA, ARIMAX, and VAR Models

For each variable we report, in order, (p,d,q) specification of multi-step ARIMA; (p,d,q) specification of single-step ARIMA; (p,d,q) specification of ARIMAX; (l) lags for VAR.

Consumption (3,0,4), (4,0,1), (4,0,4), (8); investments (2,0,2), (4,0,1), (4,0,4), (5); imports (2,0,4), (4,0,2), (0,0,4), (8); exports (6,0,4), (4,0,1), (0,0,4), (8); inflation (8,0,2), (4,0,1), (4,0,2), (8); wages (3,1,2), (2,1,1), (4,1,2), (7); per capita GDP (8,1,4), (2,1,1), (5,1,4), (7); firms' profits (2,1,2), (3,1,1), (4,1,4), (7); corporate taxation (8,1,1), (3,1,1), (7,1,4), (7); private savings (3,1,4), (3,1,1), (4,1,4), (7); expen. durable goods (3,0,4), (4,0,1), (0,0,4), (7); GDP (8,0,2), (4,0,1), (8,0,4), (5); household income (3,0,4), (3,0,2), (0,0,4), (5); government expenditure (4,0,4), (4,0,1), (4,0,4), (8); house prices (6,0,4), (4,0,1), (8,0,4), (8); employment rate (8,0,3), (4,0,1), (4,0,2), (8); unemployment rate (8,0,4), (4,0,1), (7,0,3), (8); unemp. over 27 weeks (8,0,4), (4,0,2), (4,0,4), (7).

References

1. Fan, J.; Ma, C.; Zhong, Y. A selective overview of deep learning. *Stat Sci.* **2021**, *36*, 264–290. [CrossRef] [PubMed]
2. Mathew, A.; Amudha, P.; Sivakumari, S. Deep Learning Techniques: An Overview. In *Advanced Machine Learning Technologies and Applications, Proceedings of the International Conference on Advances in Intelligent Systems and Computing (AMLTA 2020), Jaipur, India, 13–15 February 2020*; Hassanien, A., Bhatnagar, R., Darwish, A., Eds.; Springer: Singapore, 2020; Volume 1141, p. 1141. [CrossRef]
3. Chakraborty, C.; Joseph, A. *Machine Learning at Central Banks*; Working Paper 674; Bank of England: London, UK, 2017.
4. Espeholt, L.; Agrawal, S.; Sonderby, C.; Kumar, M.; Heek, J.; Bromberg, C.; Gazen, C.; Carver, R.; Andrychowicz, M.; Hickey, J.; et al. Skillful twelve hour precipitation forecasts using large context neural networks. *arXiv* **2021**, arXiv:2111.07470.
5. Chen, G.; Liu, S.; Jiang, F. Daily Weather Forecasting Based on Deep Learning Model: A Case Study of Shenzhen City, China. *Atmosphere* **2022**, *13*, 1208. [CrossRef]

6. Corsaro, S.; De Simone, V.; Marino, Z.; Scognamiglio, S. l1-Regularization in portfolio selection with machine learning. *Mathematics* **2022**, *10*, 540. [CrossRef]
7. Asawa, Y.S. Modern Machine Learning Solutions for Portfolio Selection. *IEEE Eng. Manag. Rev.* **2022**, *50*, 94–112. [CrossRef]
8. Kumbure, M.M.; Lohrmann, C.; Luukka, P.; Porras, J. Machine learning techniques and data for stock market forecasting: A literature review. *Expert Syst. Appl* **2022**, *197*, 116659. [CrossRef]
9. Khalid, A.; Huthaifa, K.; Hamzah, A.A.; Anas, R.A.; Laith, A. A New Stock Price Forecasting Method Using Active Deep Learning Approach. *J. Open Innov. Technol. Mark. Complex.* **2022**, *8*, 96, ISSN 2199-8531. [CrossRef]
10. Staffini, A. Stock Price Forecasting by a Deep Convolutional Generative Adversarial Network. *Front. Artif. Intell.* **2022**, *5*, 837596. [CrossRef]
11. Namini, S.S.; Tavakoli, N.; Namin, A.S. A Comparison of ARIMA and LSTM in Forecasting Time Series. In Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 1394–1401. [CrossRef]
12. Zulfany, E.R.; Reina, S.; Andy, E. A Comparison: Prediction of Death and Infected COVID-19 Cases in Indonesia Using Time Series Smoothing and LSTM Neural Network. *Procedia Comput. Sci.* **2021**, *179*, 982–988, ISSN 1877-0509. [CrossRef]
13. Jin, X.; Yu, X.; Wang, X.; Bai, Y.; Su, T.; Kong, J. Prediction for Time Series with CNN and LSTM. In *Proceedings of the 11th International Conference on Modelling, Identification and Control (ICMIC2019)*; Lecture Notes in Electrical Engineering; Wang, R., Chen, Z., Zhang, W., Zhu, Q., Eds.; Springer: Singapore, 2020; Volume 582. [CrossRef]
14. Jama, M. Time Series Modeling and Forecasting of Somaliland Consumer Price Index: A Comparison of ARIMA and Regression with ARIMA Errors. *Am. J. Theor. Appl. Stat.* **2020**, *9*, 143–153. [CrossRef]
15. Hall, A.S. Machine Learning Approaches to Macroeconomic Forecasting. *Econ. Rev. Fed. Reserve Bank Kans. City* **2018**, *103*, 63–81. [CrossRef]
16. Khan, M.A.; Abbas, K.; Su'ud, M.M.; Salameh, A.A.; Alam, M.M.; Aman, N.; Mehreen, M.; Jan, A.; Hashim, N.A.A.B.N.; Aziz, R.C. Application of Machine Learning Algorithms for Sustainable Business Management Based on Macro-Economic Data: Supervised Learning Techniques Approach. *Sustainability* **2022**, *14*, 9964. [CrossRef]
17. Coulombe, P.G.; Leroux, M.; Stevanovic, D.; Surprenant, S. How is machine learning useful for macroeconomic forecasting? *J. Appl. Econom.* **2022**, *37*, 920–964. [CrossRef]
18. Nosratabadi, S.; Mosavi, A.; Duan, P.; Ghamisi, P.; Filip, F.; Band, S.S.; Reuter, U.; Gama, J.; Gandomi, A.H. Data Science in Economics: Comprehensive Review of Advanced Machine Learning and Deep Learning Methods. *Mathematics* **2020**, *8*, 1799. [CrossRef]
19. Yoon, J. Forecasting of Real GDP Growth Using Machine Learning Models: Gradient Boosting and Random Forest Approach. *Comput. Econ.* **2021**, *57*, 247–265. [CrossRef]
20. Vafin, A. Forecasting macroeconomic indicators for seven major economies using the ARIMA model. *Sage Sci. Econ. Rev.* **2020**, *3*, 1–16. [CrossRef]
21. Shijun, C.; Xiaoli, H.; Yunbin, S.; Chong, Y. Application of Improved LSTM Algorithm in Macroeconomic Forecasting. *Comput. Intell. Neurosci.* **2021**, *2021*, 4471044. [CrossRef]
22. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE* **2018**, *13*, e0194889. [CrossRef]
23. Staffini, A.; Svensson, T.; Chung, U.-I.; Svensson, A.K. Heart Rate Modeling and Prediction Using Autoregressive Models and Deep Learning. *Sensors* **2021**, *22*, 34. [CrossRef]
24. Springenberg, J.; Dosovitskiy, A.; Brox, T.; Riedmiller, M. Striving for Simplicity: The All Convolutional Net. *arXiv* **2015**, arXiv:1412.6806.
25. Graves, A.; Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM networks. In Proceedings of the IEEE International Joint Conference on Neural Networks 2005, Montreal, QC, Canada, 31 July–4 August 2005; Volume 4, pp. 2047–2052. [CrossRef]
26. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
27. Yuanyuan, C.; Zhang, Y. Adaptive sparse dropout: Learning the certainty and uncertainty in deep neural networks. *Neurocomputing* **2021**, *450*, 354–361, ISSN 0925-2312. [CrossRef]
28. Bikash, S.; Angshuman, P.; Dipti, P.M. Deterministic dropout for deep neural networks using composite random forest. *Pattern Recognit. Lett.* **2020**, *131*, 205–212, ISSN 0167-8655. [CrossRef]
29. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef] [PubMed]
30. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
31. Le, Q.; Jaitly, N.; Hinton, G. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv* **2015**, arXiv:1504.00941.
32. Calin, O. *Deep Learning Architectures: A Mathematical Approach*; Springer Series in the Data Sciences; Springer International Publishing: New York, NY, USA, 2020; ISBN 978-3-030-36721-3.
33. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980v9.
34. Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv* **2017**, arXiv:1706.05098.

35. Ge, R.; Huang, F.; Jin, C.; Yuan, Y. Escaping from Saddle Points---Online Stochastic Gradient for Tensor Decomposition. *arXiv* **2015**, arXiv:1503.02101.
36. Bengio, Y. Practical recommendations for gradient-based training of deep architectures. In *Tricks of the Trade*, 2nd ed.; Volume 7700 of Theoretical Computer Science and General Issues; Springer: Berlin/Heidelberg, Germany, 2012; pp. 437–478.
37. Masters, D.; Luschi, C. Revisiting Small Batch Training for Deep Neural Networks. *arXiv* **2018**, arXiv:1804.07612.
38. Gholamy, A.; Kreinovich, V.; Kosheleva, O. *Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation*; University of Texas at El Paso Departmental Technical Reports (CS): El Paso, TX, USA, 2018.
39. Willmott, C.J.; Matsuura, K. Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance. *Clim. Res.* **2005**, *30*, 79. [CrossRef]