Journal of Computer Science

Original Research Paper

Comparative Analysis of Deep Learning Models for Multi-Step Prediction of Financial Time Series

[1]Saugat Aryal, [2]Dheynoshan Nadarajah, [3]Prabath Lakmal Rupasinghe, [3]Chandimal Jayawardena and [1]Dharshana Kasthurirathna

[1]Department of Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka
[2]Independent Researcher, Colombo, Sri Lanka
[3]Department of Computer Systems Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka

Article history
Received: 10-08-2020
Revised: 16-10-2020
Accepted: 21-10-2020

Corresponding Author:
Saugat Aryal
Department of Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka
Email: saugat.aryl@gmail.com

Abstract: Financial time series prediction has been a key topic of interest among researchers considering the complexity of the domain and also due to its significant impact on a wide range of applications. In contrast to one-step ahead prediction, multi-step forecasting is more desirable in the industry but the task is more challenging. In recent days, advancement in deep learning has shown impressive accomplishments across various tasks including sequence learning and time series forecasting. Although most previous studies are focused on applications of deep learning models for single-step ahead prediction, multi-step financial time series forecasting has not been explored exhaustively. This paper aims at extensively evaluating the performance of various state-of-the-art deep learning models for multiple multi-steps ahead prediction horizons on real-world stock and forex markets dataset. Specifically, we focus on Long-Short Term Memory (LSTM) network and its variations, Encoder-Decoder based sequence to sequence models, Temporal Convolution Network (TCN), hybrid Exponential Smoothing-Recurrent Neural Networks (ES-RNN) and Neural Basis Expansion Analysis for interpretable Time Series forecasting (N-BEATS). Experimental results show that the latest deep learning models such as N-BEATS, ES-LSTM and TCN produced better results for all stock market related datasets by obtaining around 50% less Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) scores for each prediction horizon as compared to other models. However, the conventional LSTM-based models still prove to be dominant in the forex domain by comparatively achieving around 2% less error values.

Keywords: Financial Time Series, Forecasting, Multi-Step Prediction, Deep Learning

Introduction

Financial time series forecasting has drawn significant attention among the researchers from both academia and financial industry. It is a complex domain which requires modelling of nonlinear behaviour and stochastic pattern while learning the temporal dependencies between the data. The signal to noise ratio is considerably low which contributes to the intricacy while forecasting. Researchers and stakeholders are consistently working on implementing new methodologies for improving the accuracy of predictive models due to the higher demand from the financial market. Numerous studies have been carried out in regards to both statistical and machine learning based forecasting techniques. Deep learning based models have achieved commendable results across various fields of natural language processing (Devlin *et al*., 2018; Brown *et al*., 2020), speech processing (Ogunfunmi *et al*., 2019), neural machine translation (Bahdanau *et al*., 2014; Wu *et al*., 2016), image classification (Krizhevsky *et al*., 2012) and reinforcement learning (Silver *et al*., 2017). Moreover, recent deep architectures have also demonstrated significant improvements in accuracy for time series

forecasting (Rangapuram *et al*., 2018; Salinas *et al*., 2020) and specifically in the financial setting (Yan and Ouyang, 2018; Chen *et al*., 2019) over traditional time series models. One of the key reasons for effectively handling the non-stationary nature and complications of the changing financial environment is due to its ability to learn representations through hierarchical hidden layer structure. The multi-layer architecture allows the deep models to process and analyze the complex non-linear temporal dependencies and establish proper latent representations.

In most real world applications, multi-step or multi-horizon forecasting are more valued as opposed to single or one-step ahead prediction. Long-term prediction mechanisms can help provide key insights for optimizing resource allocation and assisting the decision making process. Investors and financial firms can re-evaluate and efficiently plan their investment strategy to gain maximum profits by observing the longer predicted trajectory. Two major multi-horizon forecasting approaches have been explored lately, based on the foundation of deep learning architecture, Iterative and Direct (Sequence-to-Sequence). Iterative method deals with recursively applying one-step ahead prediction where the predicted output is fed as input for the next forecast while direct approach overcomes the shortcoming of recursive technique by forecasting the prediction vector directly at once.

While there have been numerous investigations of multi-step forecasting across diverse fields, such as, electricity load consumption (Zheng *et al*., 2017; Masum *et al*., 2018), traffic flow (Li *et al*., 2017; Lv *et al*., 2014), renewable energy production (Ghaderi *et al*., 2017), Electrocardiogram (ECG) analysis (Chauhan and Vig, 2015), very few research have focused on financial applications (Ouyang and Yin, 2018). A comprehensive analysis of the performance of sophisticated deep learning techniques for multi-step financial time series forecasting is lacking in the literature.

This work considers the most novel and relevant deep architectures and compare them in terms of performance accuracy on financial benchmarks. Specifically, we focus on LSTM and it's two variations, Bi-directional LSTM, Stacked LSTMs, Encoder-Decoder architecture, TCN, ESRNN and N-BEATS. The experiment is carried out on stock market datasets including S&P500, DJIA and NASDAQ 100 and forex markets such as EURUSD, EURGBP and EURJPY for multiple multi-steps ahead forecast horizons (2, 3, 5, 7 and 10 steps). To the best of our knowledge, such comparison of multi-step forecasting using deep learning models for financial markets has not been done yet.

The remainder of the paper is structured into four sections. Works related to multi-step forecasting and financial multi-horizon forecasting using deep learning is described in section 2. Section 3 describes the deep learning models considered for this research. In section 4 we present the details of experiments conducted and discuss the results. The conclusion, limitations and future work is provided in section 5.

Related Work

In the field of multi-horizon time series prediction, deep learning models have been employed increasingly due to their performance dominance over statistical and traditional time series models. Direct strategy used for forecasting generally consists of sequence-to-sequence (Sutskever *et al*., 2014; Cho *et al*., 2014) architecture where the encoder encodes the historical inputs to provide a compressed representation and the decoder architecture is used to generate future predictions based on the vector. The overall model is jointly trained to generate the vector of forecast for pre-defined horizon. The Multi-horizon Quantile Recurrent forecaster (MQRNN) technique (Wen *et al*., 2017) generates the hidden latent representation of historical time series using LSTM which is then fed to Multi-Layer Perceptrons (MLPs) to produce multiple quantile forecasts for multiple horizons. The authors in (Fox *et al*., 2018) propose a novel deep multi-output forecasting framework called DeepMo for predicting blood glucose trajectories. They introduce the concept of function forecasting which predicts the representation of the data in contrast to learning the distribution of future values based on the past. To complement this model, the authors also develop new architecture to model temporal dependencies and allow information propagation across the prediction window.

A new LSTM based architecture (Laptev *et al*., 2017) for extreme event forecasting at Uber is proposed which uses an autoencoder for feature extraction which is then combined using an ensembling technique and fed to LSTM based forecaster. The novel architecture provides a framework which is trained using heterogeneous time series and achieves significant improvement over traditional stacked LSTMs. Similarly, a novel Diffusion Convolutional Recurrent Neural Network (DCRNN) for traffic forecasting is introduced in (Li *et al*., 2017). The framework integrates both spatial dependency using bidirectional random walks on the directed graph and the temporal dependency using the encoder-decoder architecture. Higher-Order Tensor RNN (HOT-RNN) is presented in (Yu *et al*., 2017) to address the long-term forecasting challenges. The proposed architecture captures the higher-order nonlinear dynamics using higher-order state interactions of previous hidden states. The authors show that the proposed architecture is more expressive and accurate than standard Recurrent Neural Network (RNN) and LSTM. With recent advancements, attention-based model like in (Fan *et al*., 2019) is also

used which allows to focus on relevant and important time steps and patterns in the historical data. In addition, Transformer-based architecture like in (Li *et al*., 2019; Lim *et al*., 2019) has also been explored.

Iterative strategy in contrast operates by recursively feeding the single-step ahead forecast as future inputs to obtain multiple forecasts. Inspired by WaveNet (Oord *et al*., 2016) architecture, authors in (Borovykh *et al*., 2017) extended it for predicting financial time series achieving better results than autoregressive and LSTM based recurrent networks. The proposed model employs dilated convolutions followed by residual skip connections and uses ReLU activation function for optimizing the training time. To account for correlation between financial time series, the model takes multivariate time series as input which allows conditioning the forecast of a time series based on its own past data as well as that of other time series. Authors in (Hussein *et al*., 2016) utilized coevolutionary RNN for multi-step time series prediction using recursive technique where cooperative coevolution and Back-Propagation Through Time (BPTT) is employed for training the neural network model.

There have been studies in regard to multi-steps ahead probabilistic forecasts as well. DeepAR (Salinas *et al*., 2020) uses autoregressive recurrent neural networks to obtain a global model by training the historical data of all related time series which generates the Gaussian distribution for the forecast. Similarly, Deep State-Space Models (DSSM) (Rangapuram *et al*., 2018) follows a similar approach by exploiting recurrent neural networks to parameterize the pre-defined linear state-space model with Kalman filtering based predictive distribution.

In regards to multi-step financial time series prediction, authors in (Ouyang and Yin, 2018) extended the concept of self-organizing Autoregressive (AR) models to Varied Length Mixture models (VLM) to forecast the financial time series over multiple steps. One significant advantage of modelling such varied length models is that it allows to preserve the relationships among the input points within the forecast horizon. A comprehensive review of deep learning based financial time series prediction across various domains is presented in (Sezer *et al*., 2020). The authors observed that deep learning models performed better than machine learning models in most of the studies. Also, most researches are based on movement prediction of financial assets for short-term forecasting and the literature on multi-step price prediction is still scarce. Specifically, a detailed overview of applicability of deep models in the stock market domain is carried out in (Jiang, 2020).

In a recent study (Chatigny *et al*., 2020) related to multivariate multi-step setting, a novel variable-length attention mechanism is proposed for improving the performance of RNN based on the Dynamic Factor Graph (DFG) framework using which a new class of self-supervised generative neural architecture is also introduced. The overall model has the capacity to effectively capture temporal dependencies for multivariate time series and performs better even with limited data. Hwang (2020) used LSTM model with trainable initial hidden states which allows the model to reconstruct the abstract representation of the time series along with its parameters and forecast the future values based on the latent representation. A comparative analysis of Autoregressive Integrated Moving Average (ARIMA), LSTM and Bidirectional LSTM (BiLSTM) for various stock indices prediction is performed in (Siami-Namini *et al*., 2019) which showed that BiLSTM performs better as compared to others.

Methodology

In this section, we provide an overview of different deep learning models used in this study. We also briefly describe the multi-step prediction technique utilized for the comparative analysis purpose.

Long-Short Term Memory

LSTM networks (Hochreiter and Schmidhuber, 1997) belong to the special category of RNN family that overcomes the exploding and vanishing gradients limitation of simple RNNs (Hochreiter, 1998). By introducing an internal cell state or memory state and gating mechanisms, it can capture long-range dependencies in the data while retaining the short term memory. They have achieved state-of-the-art performance in sequence learning domain such as machine translation (Sutskever *et al*., 2014), language modeling (Sundermeyer *et al*., 2015), signal processing (Yildirim, 2018) and audio and video processing (Eck and Schmidhuber, 2002; Liu *et al*., 2019). Application of LSTM models in the financial domain (Fischer and Krauss, 2018; Heaton *et al*., 2017; Bao *et al*., 2017) have also shown promising results outperforming other traditional statistical models.

The inner working structure of an LSTM cell is shown in Fig. 1. There are three main gates in each cell that contribute to the cell state c_t: Input gate (i_t), output gate (o_t) and forget gate (f_t). The first component is the forget gate which is responsible for controlling how much of the information should be forgotten or removed from the previous cell state. It takes two inputs, output of the previous hidden state (h_{t-1}) and input of the current state (x_t) and passes them through the sigmoid activation function which outputs a vector between 0 and 1 for each value in the cell state. The 0 output for a particular value in the cell state indicates that the information is completely removed whereas 1 represents that the information is remembered.

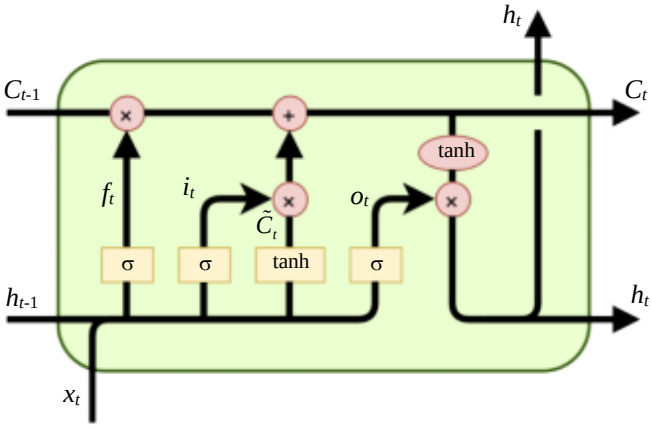

Fig. 1: LSTM cell

The input gate deals with updating the cell state with new information. It performs similar sigmoid calculation to the same set of inputs like in the forget gate and acts as a filter for the information from the previous hidden state (h_{t-1}) and current input (x_t). A candidate memory cell $\left(\tilde{C}_t\right)$ is also created to modulate the network which applies tanh activation function on the same inputs while squeezing the result in between -1 and 1. It generates a vector of all possible information that can be added to the cell state as perceived from its inputs. The negative value from tanh function indicates dropping information from the cell state while positive result infers adding new information. The output from the tanh and input gate sigmoid activation is multiplied to obtain a result which defines how much each cell state value should be updated by based on the new information:

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

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right) \quad (2)$$

$$\tilde{C}_t = \tanh\left(W_C \cdot \left[h_{t-1}, x_t\right] + b_C\right) \quad (3)$$

Finally to obtain a new cell state, the forget vector (f_t) is multiplied with the previous cell state (c_{t-1}) and the result is combined with the multiplication between input gate and tanh vector via additive operation:

$$C_t = f_t * C_{t-1} + i_t \times \tilde{C}_t \quad (4)$$

The output gate contributes in updating the hidden state (h_t) of the cell and evaluates which information from the cell state is to be used as output for the next step. The cell state vector is passed through a tanh transformation function to scale the values between -1 and +1 and then multiplied with the output vector sigmoid activation which decides whether or not the cell state value will be sent as an output for the next step and also as hidden state for the next cell:

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right) \quad (5)$$

$$h_t = o_t * \tanh\left(C_t\right) \quad (6)$$

Stacked LSTM

Deep LSTM or Stacked LSTM (Pascanu *et al*., 2013; Graves *et al*., 2013b) is an extension of the simple LSTM cell, which contains multiple LSTM cells stacked on top of each other. Adding several layers brings more depth to the architecture and increases the level of abstraction of the input sequence over time (Pascanu *et al*., 2013).

Figure 2 shows the structure of three layered stacked LSTM cells. The output from the lower hidden layer cell $\left(h_{t-1}^1\right)$ is passed as input to successive layers while each layer maintains their own hidden state and cell state.

Bidirectional LSTM

Another variation of LSTM network is Bidirectional LSTM (Graves and Schmidhuber, 2005) which processes the sequential data in both forward and backward direction using two separate hidden LSTM layers. BiLSTM connects both the layers to the same output layer. The forward layer processes the information following the same direction of the given sequence while the backward layer computes its operations using inputs from the reverse direction. Given an input sequence (x) with time steps from t-n to t-1, the hidden state of the forward layer ($\vec{h}$) traverse through the inputs from t-n to t-1, while for the backward layer the hidden state ($\bar{h}$) propagates from t-1 to t-n. Both the

layers constitute LSTM cell performing standard operations. The final output of the BiLSTM layer is given by Equation (7) where the function (σ) used to combine the two hidden states can be a concatenating, summation, average or a multiplication function:

$$y_t = \sigma\left(\vec{h}_f, \overleftarrow{h}_b\right) \tag{7}$$

The architecture of an unfolded BiLSTM layer is shown in Fig. 3. BiLSTMs have achieved great success in time series forecasting domain, such as speech recognition (Graves *et al*., 2013a) and traffic speed prediction (Cui *et al*., 2018).

Encoder-Decoder Model

Encoder-decoder architecture (Cho *et al*., 2014) or sequence-to-sequence models (Sutskever *et al*., 2014) were first introduced to overcome the limitation of RNNs to produce output sequences of arbitrary length. Since then, they have been widely used in neural machine translation (Cho *et al*., 2014; Bahdanau *et al*., 2014; Wu *et al*., 2016), speech recognition (Graves *et al*., 2013b; Chorowski *et al*., 2015; Bahdanau *et al*., 2016) and also time series forecasting tasks (Qin *et al*., 2017; Liang *et al*., 2018). In the heart of this framework lies two different networks, namely encoder and decoder where both are sequential based networks.

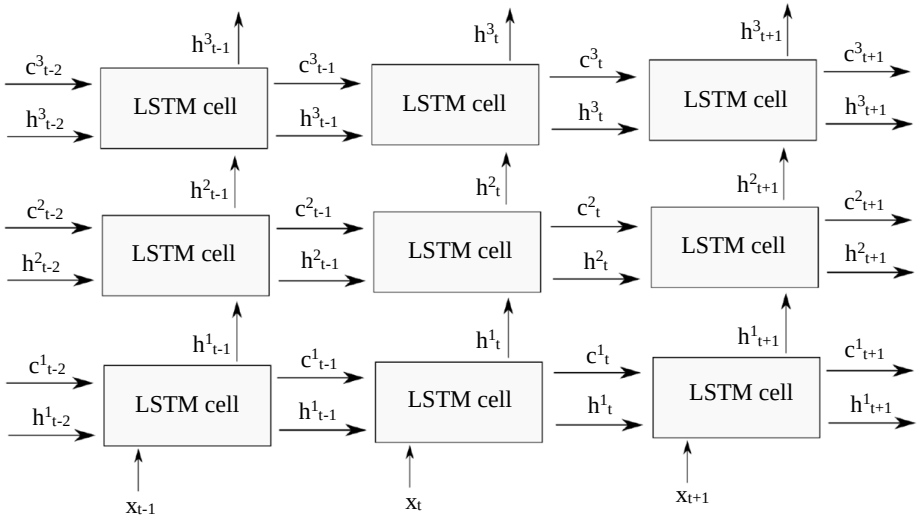

Fig. 2: Unfolded stacked LSTM

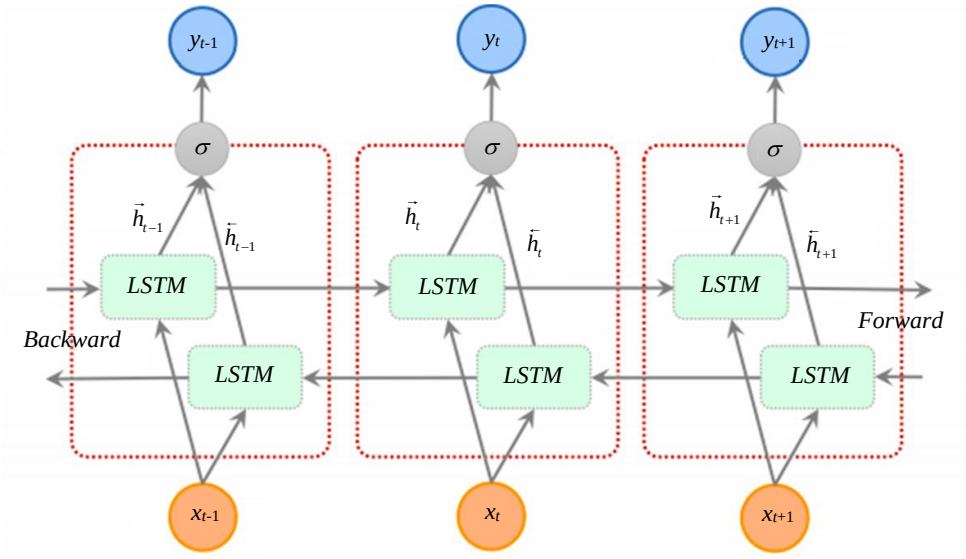

Fig. 3: Unrolled bidirectional LSTM architecture for three steps (Cui *et al*., 2018)

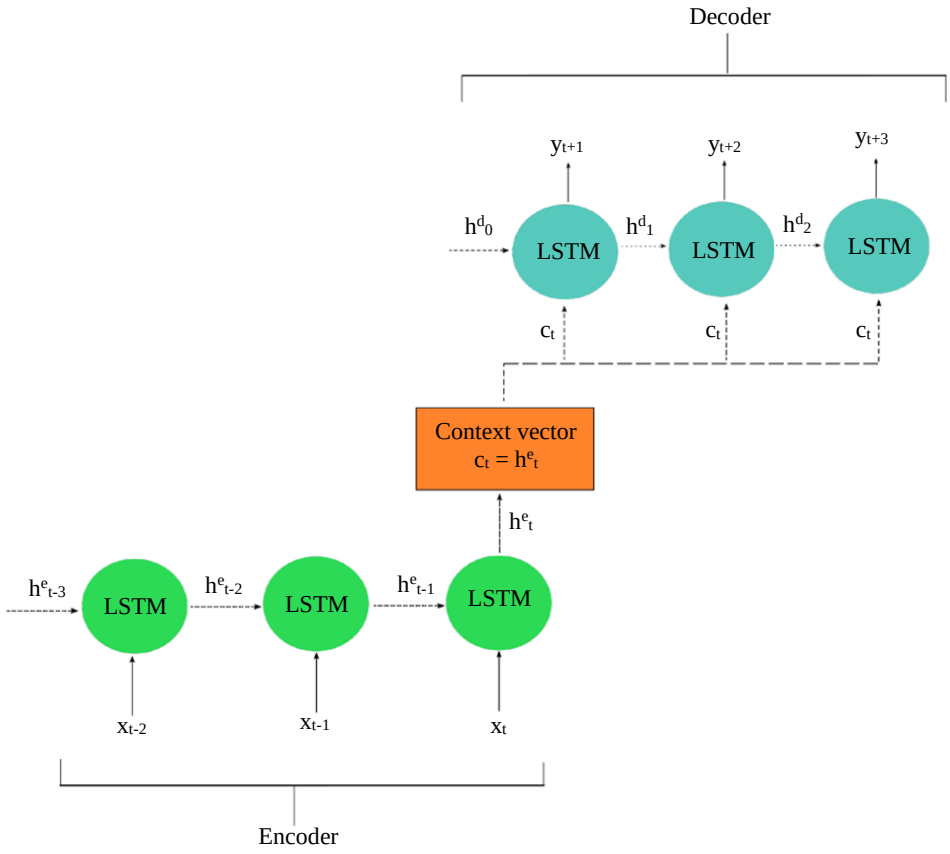

Fig. 4: Encoder-decoder model

The encoder network processes the input sequence $\vec{X}$ of length t one time step at a time and produces a fixed dimensional compressed vector representation $\vec{c}$, which is also commonly termed as context vector or latent vector and this processing of obtaining the context vector is called encoding. The context vector is usually the last hidden state ($\vec{h}_t^e$) produced from the encoder network. Then, the decoder network produces the output sequence ($\vec{y}$) given the context vector. While the decoder maintains its own hidden state, the final hidden state of the encoder network (or context vector) is replicated across each time step as inputs in a basic encoder-decoder setting. Both the encoder and decoder network can be a simple LSTM cell or stacked LSTM layers conducting its standard gating operations and are jointly trained to minimize the cost function. A general overview of the architecture is depicted in Fig. 4:

$$\vec{h}_t^e = LSTM_{encoder}\left(x_t, h_{t-1}^{\vec{e}}\right) \quad (8)$$

$$\vec{c} = \vec{h}_t^e \quad (9)$$

$$\vec{y} = LSTM_{decoder}\left(\vec{c} = \vec{h}_t^d\right) \quad (10)$$

Temporal Convolution Network

Belonging to the family of Convolutional Neural Networks (CNNs) that were initially dedicated for image dataset and computer vision tasks (Krizhevsky *et al.*, 2012; Gu *et al.*, 2018), TCN (Bai *et al.*, 2018) is an extension to adapt with sequential dataset and problems. After a series of thorough experiments, the authors claimed that TCN outperformed regular RNNs such as LSTMs on various benchmark datasets and tasks while demonstrating longer effective memory.

The basics of TCN consists of two propositions, first being that given an input sequence of arbitrary length, the network maps it to an output sequence of the same length. This principle is achieved by using 1D fully-convolutional network architecture, where the length of each hidden layer is the same as the input layer and zero padding of length (kernelsize -1) is employed such that the subsequent layers has the same length as the previous one. The second concept associated with TCN specifies that there is no information leakage from future to the past. To address this point, it replaces standard convolution operator by causal convolution such that information only from the past is used for forecasting and has no access to the future samples.

In order to achieve long-term dependencies in the sequential data and build a long effective history size, TCN makes use of dilated convolutions (Oord *et al*., 2016). It can skip outputs from the previous layer that allows to cover information from farther distance values in the sequence (increase the receptive field). Dilated convolution F on an element s of the 1-D sequence $x \in \mathbb{R}^n$ can be expressed as:

$$F(s) = \left(x *_d f\right)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{11}$$

where, d is the dilation factor and k is the filter size. Thus, the receptive field can be increased by choosing larger filter sizes k and increasing the dilation factor d. Figure 5 shows an example of a dilated causal convolution with a kernel size of 2 and a dilation factor of [1,2,4]. In addition to dilated causal convolution, TCN also implements residual blocks (He *et al*., 2016) in place of a convolutional layer to account for stabilization of deeper and larger networks. A TCN residual block consists of two layers of dilated causal convolution, weight normalization, rectified linear unit and spatial dropout. A 1×1 convolution operation is also added in each residual block to account for inconsistent input and output size.

Exponential Smoothing-Long Short Term Memory

This hybrid model (Smyl, 2020) which is an effective combination of statistical based Exponential Smoothing (ES) model and modern neural network based LSTM model is the winner of M4 competition (Makridakis *et al*., 2020) with significant margin. It is a hierarchical model which can be used to forecast multiple series, where the ES component captures the local parameters for each series such as seasonality and level whereas the weights of connections inside the LSTM model accounts for global parameters shared by all series. A high-level architecture of ES-LSTM is shown in Fig. 6. Initially, Holts-Winter exponential smoothing (Hyndman *et al*., 2008) with multiplicative seasonality is computed, however, the trend component is not accounted for as the model does not consider linear trend in the series:

$$l_t = \alpha\left(y_t / s_t\right) + (1-\alpha) l_{t-1} \tag{12}$$

$$s_{t+m} = \gamma\left(y_t / l_t\right) + (1-\gamma) s_t \tag{13}$$

where, y_t is the time series, l_t is the smoothing or level component, s is the multiplicative seasonality coefficient, m is the number of observations per seasonal period and α, γ are smoothing coefficients between zero and one.

To produce non-linear trend forecasting with multiple steps ahead, a neural network model (RNN) is used instead, the output of which is subsequently seasonalized and denormalized again to produce the forecast. Finally, the Holt-Winters model is combined with a RNN model to get the forecast from the final hybrid model:

$$\hat{y}_{t+1..t+h} = RNN\left(X_t\right) * l_t * s_{t+1..t+h} \tag{14}$$

where, h is the forecasting horizon and X_t is a vector of deseasonalized and normalized time-series derived features of which a scalar component x_t is calculated as:

$$x_t = \frac{y_t}{l_t s_t} \tag{15}$$

The neural network model employs a stack of dilated LSTM networks (Chang *et al*., 2017) interlinked with residual connections (He *et al*., 2016). Each block contains a sequence of one to four layers with each layer belonging to one of the dilated LSTM categories: Standard dilated LSTM (Chang *et al*., 2017), dual-stage attention based LSTM (Qin *et al*., 2017) and residual LSTM (Kim *et al*., 2017).

Neural Basis Expansion Analysis for Interpretable Time Series Forecasting

N-BEATS model (Oreshkin *et al*., 2019) is a pure deep neural based architecture with no time-series specific components which has achieved better forecasting accuracy than hybrid ES-RNN model on M4 competition.

At the fundamental level, the model consists of a block which is a multi-layered fully connected network with Rectified Linear Unit (ReLU) activation function that produces two outputs, the block's standard output of given horizon (forecast) and the best estimate of it's input given the functional limitations that it can operate on (backcast). The layer of blocks are combined together using a novel hierarchical doubly residual stacking topology.

Different from the common residual architecture which either involves concatenating the input of a layer to its output before passing to the subsequent layer or adding new connection from the output of each layer to the input of every other layer that follows it, the new architecture introduces two residual branches. The backcast residual branch makes it easier for subsequent blocks to forecast by removing the backcast signal from the block's input while the forecast output from each block is first integrated at the stack level and finally at the overall network level to produce the final global

forecast. The high level architecture of the model is depicted in Fig. 7.

The model is also designed to have interpretable outputs for each stack by decomposing the trend and seasonality of the series. In addition, the model is also associated with the concept of meta-learning, where the inner training loop is enclosed inside the basic building blocks while outer training procedure is contained with the parameters of the overall network, learned through gradient descent.

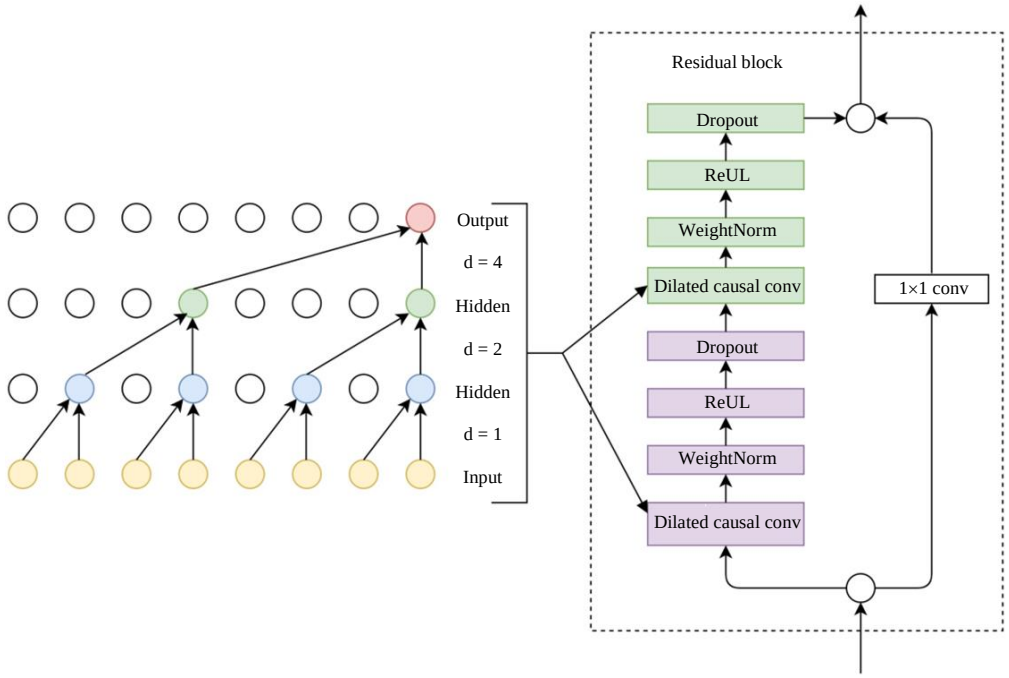

Fig. 5: TCN Architecture. The left part is the dilated causal convolution with kernel size = 2 and dilation factors d = [1,2,4]. The right part is the residual block (Qin, 2019)

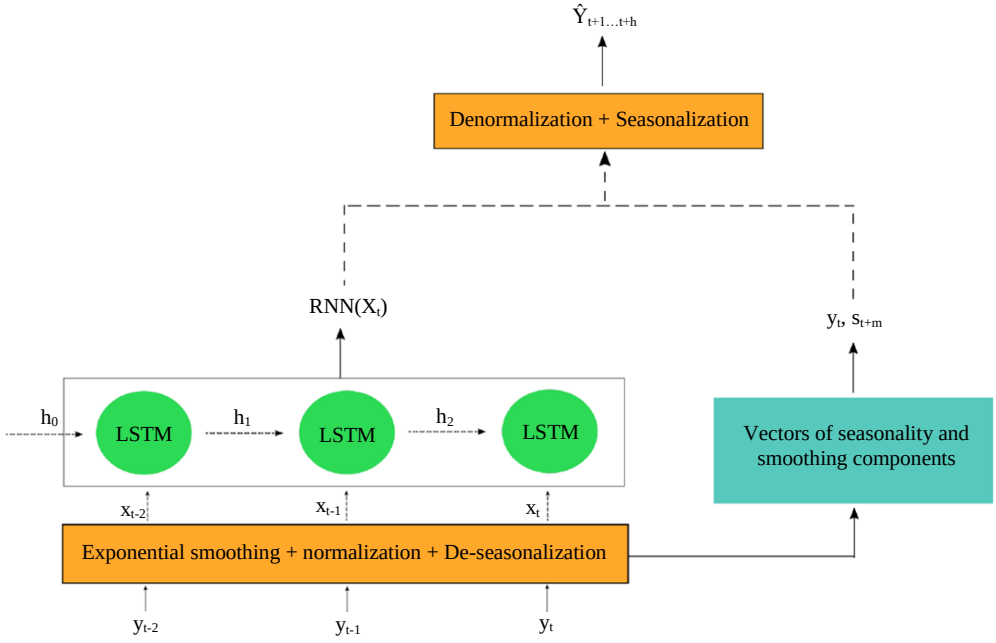

Fig. 6: ES-LSTM Architecture

DOI: 10.3844/jcssp.2020.1401.1416

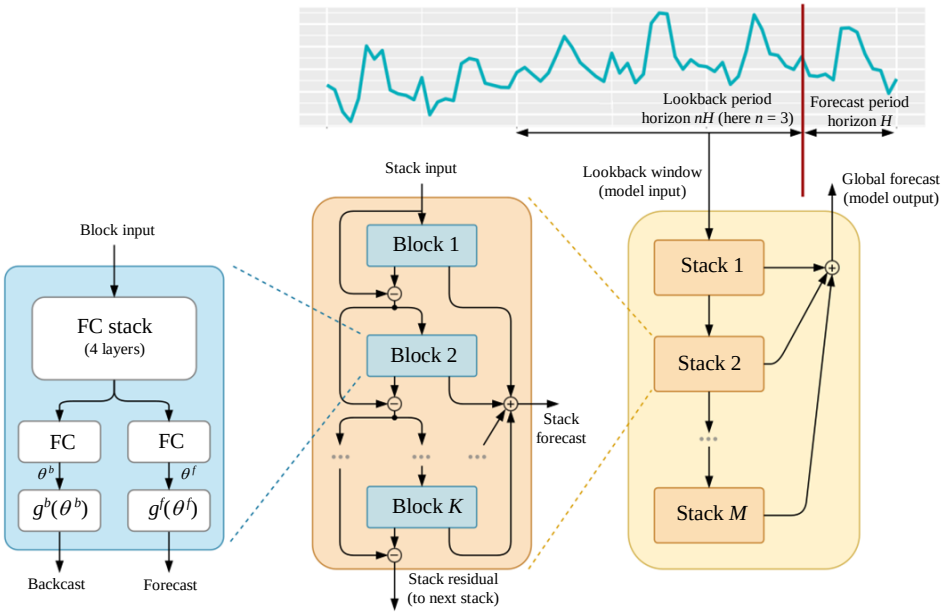

Fig. 7: N-BEATS Architecture comprising of basic block to stack to multiple stacks combined together to get final forecast (Oreshkin *et al*., 2019)

Multi-Step Prediction Strategy

While there are several methods defined in the literature for multi-step forecasting (Taieb *et al*., 2012), we focus only on Multiple-Input Multiple-Output (MIMO) (Bontempi, 2008) which has outperformed other techniques and achieved the best results for the task (Taieb *et al*., 2012). This strategy employs a single model to output the vector of future values (forecast) at one shot:

$$\hat{y}_t = F\left(x_t\right) \tag{16}$$

where, x_t is the input series vector at time t, F is the trained model and $\hat{y}_t$ is the vector of predicted output for the input sequence.

Experiments and Results

We first describe the datasets used in this study. Then, the experimental settings of different models are introduced followed by the test strategy and evaluation metrics used. Finally, we compare and analyze the performance results of various models in our benchmark datasets.

Dataset and Pre-Processing

In order to have a thorough comparison and analyze the ability of models for long-term forecast, we use six different financial benchmark datasets from the stock market and exchange rate domain. These data have been widely used in financial time series forecasting domain (Sezer *et al*., 2020). Description of the datasets is shown in Table 1. All the datasets are publicly available online and can be downloaded from the Yahoo finance website[1]. In our experiments, all the dataset has been split into training set (80%), validation set (10%) and test set (10%) in a chronological order. We perform a univariate analysis by considering only the closing price of all assets. The historical closing price values are used to predict the future values. A sliding window approach is implemented to create the supervised dataset from the training set as shown in Fig. 8. We also preprocess the data considering the large unscaled values which affects the training of the model and slows down the convergence. For each dataset, we normalize the values by subtracting the mean (μ) and dividing by the standard deviation (σ) to have 0 mean and a standard deviation of 1 as shown in Equation (17). The normalization is fit and transformed in the training set while the validation and test set is only transformed to prevent look-ahead bias:

$$\tilde{x} = \frac{x - \mu}{\sigma} \tag{17}$$

Evaluation Metrics

We use two widely adopted evaluation metrics in financial time-series forecasting domain (Guo *et al*., 2014; Sezer *et al*., 2020), Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) to test the predictive performance and efficiency of the models:

[1]https://finance.yahoo.com

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(y_t - \hat{y}_t\right)^2} \quad (18)$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}\left|y_t - \hat{y}_t\right| \quad (19)$$

where, N is the number of samples, y_t and $\hat{y}_t$ are the actual and model predicted value respectively. Both the metrics are scale-dependent measures and they represent how closer are the actual and predicted values. Hence, there is no definitive maximum value as threshold and higher values indicate less accuracy. However, values closer to zero indicate higher accuracy and better performance. While we relatively compare the error scores across different models in the same setting, the model which achieves the least scores can be defined as best performing and optimal.

Walk-Forward Validation

The test dataset is evaluated using the Walk-Forward validation sliding window approach. In this method, we take the first n values from the test set, where n is the input time lag, for which the model predicts the next h future values at once. The window shifts one step towards the right taking the actual values to predict again. This process continues until the end of the test set. In this manner, the model always predicts using the available true data. In our case where we forecast for multiple steps ahead, h = [2,3,5,7,10]. This process is similar to the sliding window approach in Fig. 8.

The RMSE and MAE is calculated at each sliding instance and finally, the average is calculated for the overall test set. Finally, we compare the average RMSE and MAE for each model and each horizon.

Experimental Details

While training the model, we have several parameters to be defined for each model. As a general setting to all the models, the batch size is set to 32 and mean squared error is selected as the loss function. We adopt Adam (Kingma and Ba, 2014) as the optimization algorithm with the learning rate set to 0.001. All the models are trained for 1000 epochs with early stopping implemented as a callback function to prevent overfitting. Specifically, we monitor the validation loss after the end of each epoch and the training process is stopped if the loss does not improve for 50 iterations. Based on the experiments conducted, we selected the input sliding window size (t) to be 16 days which depicts the best trade-off between performance accuracy and system requirements.

Table 1: Description of dataset

Domain	Dataset	Period	No. of instances
Stock market	S&P 500 index	1999-01-04 to 2019-12-30	5,282
	DJIA index	1999-01-04 to 2019-12-30	5,282
	NASDAQ	100 Index 2000-03-20 to 2019-12-30	4,983
Forex market	EURUSD	2003-12-01 to 2019-12-31	4168
	EURGBP	2003-01-01 to 2019-12-31	4,406
	EURJPY	2003-01-23 to 2019-12-31	4,373

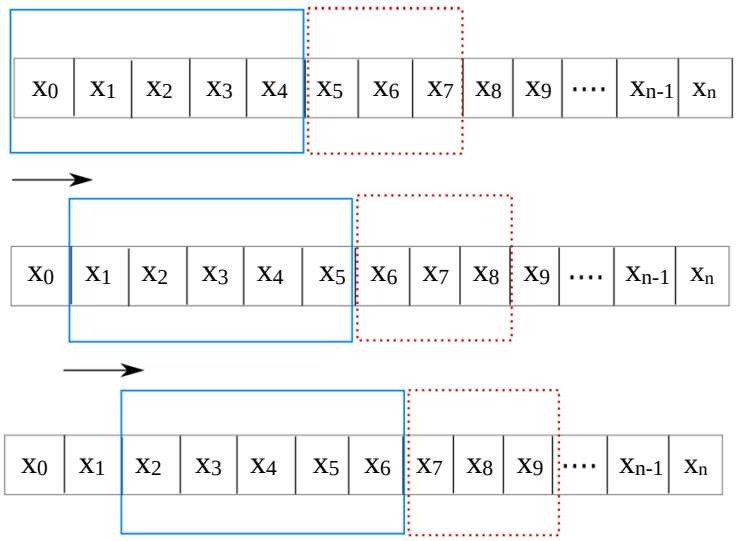

Fig. 8: Sliding window approach

All the LSTM based models adopt 100 hidden units. The simple LSTM and BiLSTM model adopts a single hidden layer while the deep LSTM model implements three layers, stacked on top of each other. Both the encoder and decoder network consist of a single layer of LSTM model with the same hidden units to obtain a compressed representation and generate the output vector. In the ESLSTM network, the seasonality was empirically selected to 30 for all the financial assets and follows the same LSTM configuration with one layer and 100 neurons.

For the TCN model, the dilations is specified to [1,2,4,8] with kernel size of 2 and the amount of filters used is set to 100 to have common grounds with the LSTM models. The model also employs a single stack of residual blocks and allows for skip connections from input to each block. With the configured parameters, the receptive field of the TCN network is the same as the selected input window size of 16.

In the N-BEATS architecture, the backcast length is set to 16 for each forecast lengths. For each stack, two blocks are considered whereas the hidden layer units in each block is set to 100. Moreover, generic architecture based stack types are used which do not depend on specific knowledge related to time-series.

All the models were implemented based on Keras library with Tensor flow backend and the experiments were conducted on a machine with Intel (R) Core (TM) i7-9700F CPU and Nvidia GeForce RTX 2080 Ti GPU. The models were trained multiple times to address the random initialization of weights and the average performance on the test set was recorded for comparison.

Results and Discussion

Our experimental results for six different financial datasets on the test dataset are shown in Table 2 to 7. Each table summarizes the average RMSE and MAE scores for multi-step continuous forecasts across several deep learning models used in this study. The error values are computed after post-processing where the model predictions are re-scaled to the original range of actual value. The stock index values are in much higher range and suffer from significant price movements as compared to the forex values. Hence, we can observe substantial difference in RMSE and MAE scores for stock and forex datasets. The best metrics scores for each forecast horizon are highlighted in bold.

The results show that the deep neural models depict inconsistent performances depending on the domain and behaviour of the financial markets. For S&P 500 index, the temporal convolutional based model drastically reduces the RMSE score by more than 50% for all forecast horizons. The ES-LSTM and pure deep neural based N-BEATS models also show significant improvements. The deep LSTM model exhibited poor results followed by simple LSTM. BiLSTM based architecture beats encoder-decoder on short-term forecasts but fails when the prediction horizon is longer (7 and 10 days).

The LSTM based models along with Encoder-Decoder architecture also performed poorly when applied to DJIA index with 57% higher error rate. Although the best performing model in this case was ES-LSTM, the RMSE and MAE scores for TCN and N-BEATS were also relatively lower. However, it can be noted that the range of errors in case of DJIA stock index is much higher in case of classical LSTM models as compared to S&P 500. Based on the results, it is also worth commenting that simple LSTM and BiLSTM outperform sequence to sequence architecture that are designed to capture the complex temporal relationship in an effective way.

Similar to the other two stock markets, deep LSTM recorded the highest error scores compared to other models for NASDAQ 100. Also, TCN architecture outperformed other models for all prediction horizons except for 7 days ahead for which ES-LSTM obtained better result. N-BEATS also exhibited considerable accuracy compared to the other models. With regards to memory based models, the error metrics for BiLSTM is relatively lower as compared to LSTM and Encoder-Decoder.

Overall, we can observe that the state-of-the-art TCN, ES-LSTM and N-BEATS architectures heavily outperformed other traditional models in the stock market domain for almost all forecast horizons. We can also note that the RMSE and MAE scores do not always gradually increase along with the forecast horizons. This is consistent with previous research findings (Bao *et al*., 2014) while forecasting chaotic time series for multiple horizons.

The forex datasets behave differently as compared to the stock indexes. LSTM based models prove to be more dominant and display better results in most cases. N-BEATS recorded the least error for short-term and long-range forecast horizons (2,3 and 10 days) for EURUSD. However, deep LSTM outperformed others in case of mid-range (5 and 7 days) forecasts. Similarly, LSTM, stacked LSTM, BiLSTM and Encoder-decoder architectures performed better for most forecast horizons for the Euro to Pounds market (EURGBP). In some scenarios, we can report that multiple models achieve similar accuracy for same forecast horizons such as, stacked LSTM and BiLSTM achieved the same score for 2 days ahead forecast. Also, both encoder-decoder and TCN model recorded least RMSE and MAE scores while forecasting 5 days ahead. The stacked-LSTM exhibits the least error score for 3, 5 and 7 days ahead forecast of EURJPY exchange rate, while BiLSTM and simple LSTM outperforms other models while forecasting 2 and 10 days ahead respectively. Unlike other two forex dataset, RMSE and MAE scores are relatively higher for all models in case of long-range forecast (10 days) of Euro to Yen. It is interesting to mention that ES-LSTM

was the least accurate model for all three forex markets. Similarly, unlike stock market datasets, almost all models follow gradual linear trend as the forecast horizon increases.

Table 2: S&P 500 Comparison Results

		Forecast horizon				
Models	Metrics	2 Days	3 Days	5 Days	7 Days	10 Days
LSTM	RMSE	114.840	125.333	178.050	224.790	325.602
	MAE	113.113	110.433	174.615	220.959	321.152
Stacked-LSTM	RMSE	151.301	200.239	529.958	414.273	555.635
	MAE	147.466	191.411	528.542	411.932	553.821
BiLSTM	RMSE	63.316	58.478	96.854	149.500	246.797
	MAE	56.318	46.648	81.689	133.670	238.542
Encoder-Decoder	RMSE	103.707	120.679	88.291	125.773	199.286
	MAE	93.327	111.369	80.764	114.835	187.693
ES-LSTM	RMSE	43.887	44.267	86.6217	65.301	81.287
	MAE	41.489	40.138	76.738	58.628	70.002
TCN	RMSE	**24.617**	**29.955**	**36.251**	**42.306**	**48.876**
	MAE	**22.869**	**26.574**	**31.748**	**36.709**	**41.985**
N-BEATS	RMSE	30.236	35.069	54.695	56.054	59.727
	MAE	28.320	32.229	47.619	47.963	51.946

Table 3: DJIA comparison results

		Forecast horizon				
Models	Metrics	2 Days	3 Days	5 Days	7 Days	10 Days
LSTM	RMSE	1698.271	1511.102	2399.312	2420.768	2363.74
	MAE	1691.548	1497.159	2380.754	2395.543	2333.575
Stacked-LSTM	RMSE	4513.165	5869.620	7365.875	5957.295	6347.864
	MAE	4506.461	5850.385	7361.250	5951.215	6336.358
BiLSTM	RMSE	2767.882	3130.423	3058.695	3802.990	3201.650
	MAE	2729.544	3117.111	2896.899	3694.886	3177.442
Encoder-Decoder	RMSE	3655.777	3742.156	3805.998	3130.119	4059.851
	MAE	3647.535	3737.042	3794.176	3092.931	4038.078
ES-LSTM	RMSE	**43.887**	**44.267**	**86.621**	**65.301**	**81.287**
	MAE	**41.489**	**40.138**	**76.738**	**58.628**	**70.002**
TCN	RMSE	236.790	295.575	355.070	425.817	471.371
	MAE	217.392	265.053	310.963	371.231	406.477
N-BEATS	RMSE	364.098	463.798	832.432	611.659	1506.272
	MAE	314.700	405.783	665.600	501.405	1286.235

Table 4: NASDAQ 100 comparison results

		Forecast horizon				
Models	Metrics	2 Days	3 Days	5 Days	7 Days	10 Days
LSTM	RMSE	423.847	780.648	1992.599	682.167	705.546
	MAE	382.708	777.880	1988.240	675.050	694.545
Stacked-LSTM	RMSE	973.453	1548.454	2095.927	2306.527	2534.892
	MAE	970.076	1504.349	2059.833	2257.059	2515.681
BiLSTM	RMSE	377.478	552.569	416.247	460.123	719.576
	MAE	364.704	548.040	406.244	450.262	709.804
Encoder-Decoder	RMSE	467.282	446.732	1025.974	1144.887	889.599
	MAE	409.260	386.498	1018.017	1128.694	844.893
ES-LSTM	RMSE	104.654	307.713	213.457	**159.068**	401.948
	MAE	96.044	282.699	195.484	**140.833**	377.301
TCN	RMSE	**88.682**	**163.101**	**115.347**	227.868	**155.766**
	MAE	**80.513**	**153.098**	**101.124**	192.354	**135.187**
N-BEATS	RMSE	222.861	238.118	259.671	344.024	247.899
	MAE	218.531	231.058	233.568	333.954	229.427

Table 5: EURUSD comparison results

		Forecast horizon				
Models	Metrics	2 Days	3 Days	5 Days	7 Days	10 Days
LSTM	RMSE	0.00427	0.00497	0.00607	0.00674	0.00753
	MAE	0.00395	0.00448	0.00536	0.00592	0.00654
Stacked-LSTM	RMSE	0.00433	0.00532	**0.00587**	**0.00668**	0.00836
	MAE	0.00402	0.00486	**0.00520**	**0.00589**	0.00744
BiLSTM	RMSE	0.00423	0.00492	0.00593	0.00702	0.00744
	MAE	0.00390	0.00443	0.00523	0.00623	0.00646
Encoder-Decoder	RMSE	0.00415	0.00487	0.00593	0.00722	0.00791
	MAE	0.00385	0.00440	0.00525	0.00637	0.00691
ES-LSTM	RMSE	0.00504	0.00823	0.00780	0.00791	0.00854
	MAE	0.00454	0.00747	0.00692	0.00693	0.00739
TCN	RMSE	0.00418	0.00497	0.00618	0.00693	0.00772
	MAE	0.00384	0.00447	0.00541	0.00605	0.00668
N-BEATS	RMSE	**0.00412**	**0.00480**	0.00631	0.00735	**0.00738**
	MAE	**0.00378**	**0.00430**	0.00539	0.00643	**0.00641**

Table 6: EURGBP comparison results

		Forecast horizon				
Models	Metrics	2 Days	3 Days	5 Days	7 Days	10 Days
LSTM	RMSE	0.00377	**0.00443**	0.00558	0.00664	0.00779
	MAE	0.00347	**0.00397**	0.00489	0.00573	0.00670
Stacked-LSTM	RMSE	**0.00372**	0.00444	0.00546	0.00640	0.00774
	MAE	**0.00344**	0.00400	0.00482	0.00558	0.00668
BiLSTM	RMSE	**0.00372**	0.00452	0.00553	0.00673	0.00817
	MAE	**0.00344**	0.00400	0.00483	0.00585	0.00705
Encoder-Decoder	RMSE	0.00407	0.00454	**0.00537**	**0.00609**	**0.00714**
	MAE	0.00377	0.00408	**0.00471**	**0.00528**	**0.00618**
ES-LSTM	RMSE	0.00386	0.00733	0.00970	0.01452	0.00849
	MAE	0.00355	0.00694	0.00893	0.01363	0.00756
TCN	RMSE	0.00379	0.00445	**0.00537**	0.00620	0.00773
	MAE	0.00351	0.00400	**0.00471**	0.00542	0.00676
N-BEATS	RMSE	0.00395	0.00463	0.00572	0.00691	0.00997
	MAE	0.00367	0.00416	0.00503	0.00600	0.00844

Table 7: EURJPY comparison results

		Forecast horizon				
Models	Metrics	2 Days	3 Days	5 Days	7 Days	10 Days
LSTM	RMSE	0.56835	0.69033	0.85884	0.99597	**1.15732**
	MAE	0.52265	0.61943	0.76053	0.86914	**1.00114**
Stacked-LSTM	RMSE	0.57771	**0.66684**	**0.85445**	**0.98912**	1.26894
	MAE	0.53376	**0.59929**	**0.75537**	**0.86162**	1.11844
BiLSTM	RMSE	**0.56197**	0.67186	0.88569	1.01427	1.17914
	MAE	**0.51602**	0.60178	0.78782	0.88905	1.02423
Encoder-Decoder	RMSE	0.56418	0.68552	0.88250	1.04270	1.19979
	MAE	0.52066	0.61837	0.77768	0.91007	1.04313
ES-LSTM	RMSE	0.58944	0.72175	0.97596	1.07270	1.22054
	MAE	0.54230	0.65220	0.87310	0.94332	1.05797
TCN	RMSE	0.59088	0.69200	0.90328	1.11289	1.27454
	MAE	0.54047	0.61870	0.79469	0.96318	1.10717
N-BEATS	RMSE	0.59328	0.71860	0.88381	1.07301	1.22428
	MAE	0.54714	0.65091	0.78455	0.93912	1.06629

Analysis of results show that sophisticated deep models provides promising avenue in effectively capturing the underlying dynamics and patterns of the stock market. Specifically, the inherent hierarchical learning ability of N-BEATS and TCN architecture as well as the pre-processed exponential smoothing combined with LSTM model allows for remarkable results. However, the gated mechanism based pure

LSTM, Bidirectional LSTM and the sequential architecture outperforms the state-of-the-art models in forex market with small margin.

Conclusion

In this study, we investigated the performance of the most relevant deep learning models for multi-step-ahead forecasts in the financial domain. The experiments have been carried out on three real world datasets of the stock market and forex domain. After a sound comprehensive assessment, we can observe that the recent benchmark models for time-series prediction such as ES-LSTM, TCN and N-BEATS, showcased exceptional results for the stock market domain, while the classical LSTM, BiLSTM and Encoder-Decoder models still have an upperhand in predicting the forex markets. The results obtained also conclude that the relationship between forecast horizon and error values is non-linear for most of the models in the stock domain, while the forex market manifests linear correlation for almost all neural networks.

This research is however limited to univariate analysis where only historical data is considered as the source of input. The stochastic and dynamically driven financial field is significantly impacted by several other external factors such as news (Du and Tanaka-Ishii, 2020) and interrelationship between multiple time series (Borovykh *et al*., 2017) which could be incorporated and examined in the future work. Also, hyper parameter tuning for each model can be automated using an optimization algorithm (Bergstra *et al*., 2011).

Acknowledgement

This research is carried out in collaboration with and funded by Capital Alliance Limited[2].

Author's Contributions

All the authors have equal contribution for the completion of the manuscript.

Ethics

This research work is original and contains unpublished material. All the authors have read and approved the manuscript and no ethical issues are involved with no conflict of interest to disclose.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., & Bengio, Y. (2016, March). End-to-end attention-based large vocabulary speech recognition. In 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 4945-4949). IEEE.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 12(7), e0180944.

Bao, Y., Xiong, T., & Hu, Z. (2014). Multi-step-ahead time series prediction using multiple-output support vector regression. Neurocomputing, 129, 482-493.

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In Advances in neural information processing systems (pp. 2546-2554).

Bontempi, G. (2008). Long term time series prediction with multi-input multi-output local learning. Proc. 2nd ESTSP, 145-154.

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. arXiv preprint arXiv:1703.04691.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Agarwal, S. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., ... & Huang, T. S. (2017). Dilated recurrent neural networks. In Advances in Neural Information Processing Systems (pp. 77-87).

Chatigny, P., Patenaude, J. M., & Wang, S. (2020). Financial Time Series Representation Learning. arXiv preprint arXiv:2003.12194.

Chauhan, S., & Vig, L. (2015, October). Anomaly detection in ECG time signals via deep long short-term memory networks. In 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA) (pp. 1-7). IEEE.

Chen, L., Chi, Y., Guan, Y., & Fan, J. (2019, May). A hybrid attention-based EMD-LSTM model for financial time series prediction. In 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD) (pp. 113-118). IEEE.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. In Advances in neural information processing systems (pp. 577-585).

[2]https://cal.lk/

Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Du, X., & Tanaka-Ishii, K. (2020, July). Stock Embeddings Acquired from News Articles and Price History and an Application to Portfolio Optimization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 3353-3363).

Eck, D., & Schmidhuber, J. (2002, September). Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In Proceedings of the 12th IEEE workshop on neural networks for signal processing (pp. 747-756). IEEE.

Fan, C., Zhang, Y., Pan, Y., Li, X., Zhang, C., Yuan, R., ... & Huang, H. (2019, July). Multi-horizon time series forecasting with temporal attention learning. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2527-2535).

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

Fox, I., Ang, L., Jaiswal, M., Pop-Busui, R., & Wiens, J. (2018, July). Deep multi-output forecasting: Learning to accurately predict blood glucose trajectories. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 1387-1395).

Ghaderi, A., Sanandaji, B. M., & Ghaderi, F. (2017). Deep forecast: Deep learning-based spatio-temporal forecasting. arXiv preprint arXiv:1707.08110.

Graves, A., Jaitly, N., & Mohamed, A. R. (2013a, December). Hybrid speech recognition with deep bidirectional LSTM. In 2013 IEEE workshop on automatic speech recognition and understanding (pp. 273-278). IEEE.

Graves, A., Mohamed, A. R., & Hinton, G. (2013b, May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). IEEE.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural networks, 18(5-6), 602-610.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern Recognition, 77, 354-377.

Guo, Z., Wang, H., Liu, Q., & Yang, J. (2014). A feature fusion based forecasting model for financial time series. PloS one, 9(6), e101113.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: deep portfolios. Applied Stochastic Models in Business and Industry, 33(1), 3-12.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6, 107-116.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Hussein, S., Chandra, R., & Sharma, A. (2016, July). Multi-step-ahead chaotic time series prediction using coevolutionary recurrent neural networks. In 2016 IEEE Congress on Evolutionary Computation (CEC) (pp. 3084-3091). IEEE.

Hwang, J. (2020). Modeling Financial Time Series using LSTM with Trainable Initial Hidden States. arXiv preprint arXiv:2007.06848.

Hyndman, R., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). Forecasting with exponential smoothing: the state space approach. Springer Science & Business Media.

Jiang, W. (2020). Applications of deep learning in stock market prediction: recent progress. arXiv preprint arXiv:2003.01859.

Kim, J., El-Khamy, M., & Lee, J. (2017). Residual LSTM: Design of a deep recurrent architecture for distant speech recognition. arXiv preprint arXiv:1701.03360.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Laptev, N., Yosinski, J., Li, L. E., & Smyl, S. (2017, August). Time-series extreme event forecasting with neural networks at uber. In International Conference on Machine Learning (Vol. 34, pp. 1-5).

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y. X., & Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Advances in Neural Information Processing Systems (pp. 5243-5253).

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926.

Liang, Y., Ke, S., Zhang, J., Yi, X., & Zheng, Y. (2018, July). Geoman: Multi-level attention networks for geo-sensory time series prediction. In IJCAI (pp. 3428-3434).

Lim, B., Arik, S. O., Loeff, N., & Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting. arXiv preprint arXiv:1912.09363.

Liu, A. A., Shao, Z., Wong, Y., Li, J., Su, Y. T., & Kankanhalli, M. (2019). LSTM-based multi-label video event detection. Multimedia Tools and Applications, 78(1), 677-695.

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. Y. (2014). Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2), 865-873.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. International Journal of Forecasting, 36(1), 54-74.

Masum, S., Liu, Y., & Chiverton, J. (2018, June). Multi-step time series forecasting of electric load using machine learning models. In International Conference on Artificial Intelligence and Soft Computing (pp. 148-159). Springer, Cham.

Ogunfunmi, T., Ramachandran, R. P., Togneri, R., Zhao, Y., & Xia, X. (2019). A primer on deep learning architectures and applications in speech processing. Circuits, Systems and Signal Processing, 38(8), 3406-3432.

Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437.

Ouyang, Y., & Yin, H. (2018). Multi-step time series forecasting with an ensemble of varied length mixture models. International journal of neural systems, 28(04), 1750053.

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026.

Qin, H. (2019). Comparison of Deep learning models on time series forecasting: a case study of Dissolved Oxygen Prediction. arXiv preprint arXiv:1911.08414.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. arXiv preprint arXiv:1704.02971.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. In Advances in neural information processing systems (pp. 7785-7794).

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting, 36(3), 1181-1191.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied Soft Computing, 90, 106181.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). A comparative analysis of forecasting financial time series using arima, lstm and bilstm. arXiv preprint arXiv:1911.09512.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Chen, Y. (2017). Mastering the game of go without human knowledge. nature, 550(7676), 354-359.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. International Journal of Forecasting, 36(1), 75-85.

Sundermeyer, M., Ney, H., & Schlüter, R. (2015). From feedforward to recurrent LSTM neural networks for language modeling. IEEE/ACM Transactions on Audio, Speech and Language Processing, 23, 517-529.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert systems with applications, 39(8), 7067-7083.

Wen, R., Torkkola, K., Narayanaswamy, B., & Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. arXiv preprint arXiv:1711.11053.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Yan, H., & Ouyang, H. (2018). Financial time series prediction based on deep learning. Wireless Personal Communications, 102(2), 683-700.

Yildirim, Ö. (2018). A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification. Computers in biology and medicine, 96, 189-202.

Yu, R., Zheng, S., Anandkumar, A., & Yue, Y. (2017). Long-term forecasting using higher order tensor RNNs. arXiv preprint arXiv:1711.00073.

Zheng, J., Xu, C., Zhang, Z., & Li, X. (2017, March). Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network. In 2017 51st Annual Conference on Information Sciences and Systems (CISS) (pp. 1-6). IEEE.