DeepTrace: A Generic Framework for Time Series Forecasting

Nithish B. Moudhgalya[1(✉)], Siddharth Divi[1], V. Adithya Ganesan[1], S. Sharan Sundar[1], and Vineeth Vijayaraghavan[2]

[1] Sri Sivasubramaniya Nadar College of Engineering, Chennai, India
{nithish.moudhgalya,siddharthdivi,v.adithyaganesan, sharan.sundar}@ieee.org
[2] Solarillion Foundation, Chennai, India
vineethv@ieee.org

Abstract. We propose a generic framework for time-series forecasting called DeepTrace, which comprises of 5 model variants. These variants are constructed using two or more of three task specific components, namely, Convolutional Block, Recurrent Block and Linear Block, combined in a specific order. We also introduce a novel training methodology by using future contextual frames. However, these frames are dropped during the testing phase to verify the robustness of DeepTrace in real-world scenarios. We use an optimizer to offset the loss incurred due to the non-provision of future contextual frames. The genericness of the framework is tested by evaluating the performance on real-world time series datasets across diverse domains. We conducted substantial experiments that show the proposed framework outperforms the existing state-of-art methods.

Keywords: Time Series · Deep framework · Bidirectional

1 Introduction

Time Series forecasting has a wide range of applications across domains like finance (stocks prediction), economics (interest rates of central bank, monthly changes to macro-economic indicators like GDP), engineering (speech and audio signals, video analysis, image recognition), natural sciences (temperature, seismic signals) and neuroscience (brain activity measurements via EEG and fMRI), to name a few. Several approaches have been used in time series modeling. Traditionally, statistical models include linear forecasting methods like moving average, exponential smoothing, and autoregressive integrated moving average (ARIMA). Because of their relative simplicity, linear models have been the main research focus during the past few decades. Lately, to overcome the limitations of the linear models and to account for the nonlinear patterns observed in real-world problems, several classes of nonlinear models like the bilinear model [1], the autoregressive conditional heteroscedastic (ARCH) model [2] and the threshold

I. Rojas et al. (Eds.): IWANN 2019, LNCS 11506, pp. 139–151, 2019.
https://doi.org/10.1007/978-3-030-20521-8_12

autoregressive (TAR) model [3] have been proposed. Although non-linear models show significant improvement, using them as general forecasting solutions is still limited [4]. With the advent of artificial neural networks and deep learning concepts that showcase flexible nonlinear modeling capability, time series forecasting has seen a massive enhancement.

Currently, recurrent neural networks (RNNs), and in particular the long-short term memory unit (LSTM) are referred to as the state-of-art solutions for time series forecasting [5,6]. The efficiency of these networks can be explained by the recurrent connections that allow the network to access previous time series values. Recently, convolutional neural networks (CNN) have also been widely used for time series forecasting because of their relative computational efficiency and ability to effectively capture local patterns. Convolutions with auto regressive properties [7], multiple layers of dilated convolutions [8] were used for time series forecast and classification. Multi-scale convolutional neural network (MCNN) [9] and fully convolutional network (FCN) [10] are deep learning approaches that take advantage of CNN for end-to-end classification of univariate time series. However, in all these works, the proposed algorithms and models are specific to the use case and dataset considered.

In this paper, a generic framework is proposed for time series forecasting, using a novel training methodology that involves providing future contextual frames (refer Sect. 2.3) during training. The framework achieves state-of-art results across diverse domains of time series datasets (mentioned in Sect. 6). We begin by defining some basic concepts, definitions and formulae. We then describe the components used in the framework, followed by the various architectural variants. Following this, we describe the novel training methodology used and the results of the proposed framework across 3 diverse datasets. In the end, we conclude with some inferences drawn from the results of the architectural variants.

2 Preliminary

2.1 Autocorrelation

Autocorrelation is the Pearson correlation coefficient calculated between two values of the same variable at times X_t and X_{t+k}, where k denotes the delay value. It is calculated as

$$R(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2} \tag{1}$$

Where μ is the mean of the time series, σ is its standard deviation and E refers to expectation. According to Eq. (1), a value close to $+1$ signifies a positively correlated data, a value close to -1 signifies a negatively correlated data and that with 0 has no correlation. Autocorrelation at $k = 0$ is always $+1$ as a time series is always perfectly correlated with itself.

2.2 1-D Convolutions

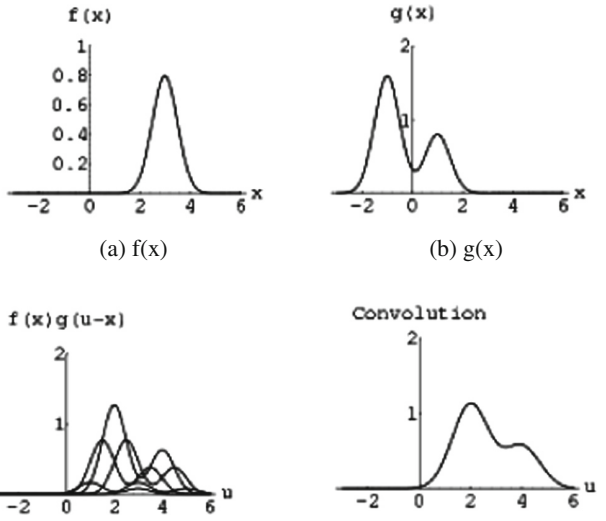
Convolution is mathematically defined as an integral that expresses the amount of overlap of one function $g(x)$ as it is shifted over another function $f(x)$ i.e, common properties and characteristics of both the functions.

$$h(x) = f(x) * g(x) = g(x) * f(x)$$
$$h(x) = \int_{-\infty}^{\infty} f(u).g(u-x)du \tag{2}$$

The Fig. 1d below depicts the outcome of applying a convolution function on $g(x)$ using $f(x)$ shown in Fig. 1b and a respectively. In Convolutional Neural Networks (CNN), inputs are analogous to the function $g(x)$ and kernels are analogous to $f(x)$. Each layer in a CNN operates with n number of kernels $\{f_1(x), f_2(x), ... f_n(x)\}$ on the input data $g(x)$.

2.3 Data Preparation

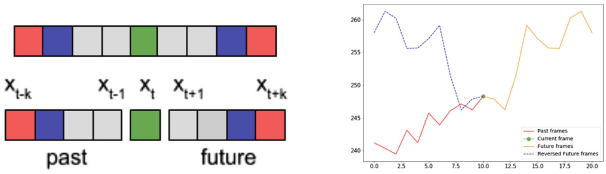
The data fed into the model is a 3-dimensional structure with dimensions as number of samples, time steps and the features. The data cube thus created can be visualized as a temporally increasing order from left to right as shown in Fig. 2a. If x_t is the current data frame, the data would be segmented into 2 sets of frames - x_{t-k} to x_{t-1} and x_{t+1} to x_{t+k}, see Fig. 2a. The former segment is referred as the past and the latter as the future frames. The variable k refers to the span of the contextual vectors, i.e. the parameter that controls the amount of contextual information fed to the model.



(a) f(x) (b) g(x)

(c) Conv. of $f(x)$ and $g(x)$ (d) Integrated convolution fn.

Fig. 1. 1D - Convolution



(a) Data Preparation Format (b) Reversing Future Frames

Fig. 2. Data preparation

It is conclusive from the works of [11], that providing reversed sequences provides further context to the model. From Fig. 2b, it can be seen that both the past and reversed future data frames converge from $t-k$ and $t+k$ to t respectively. Hence the future frames were flipped along the temporal axis providing data frames ranging from x_{t+k} to x_{t+1}. The past frames are henceforth referred to as P and the future frames as F. By convention, F' represents the reversed future frames.

2.4 Metrics

We use the following metrics to measure and compare the performance of DeepTrace across different domains of datasets.

$$\textbf{Coefficient of Determination}(R^2) = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \mu_y)^2}$$

$$\textbf{Mean Squared Error}(MSE) = \sum_i (f_i - y_i)^2$$

$$\textbf{Correlation}(CORR) = \frac{E|(f_i - \mu_f) * (y_i - \mu_y)|}{\sigma_f \sigma_y}$$

$$\textbf{Root Relative Squared Error}(RSE) = \frac{\sqrt{\sum_i (y_i - f_i)^2}}{\sqrt{\sum_i (y_i - \mu_y)^2}}$$

Where,
y_i represents truth values
f_i represents model predictions
μ_y represents mean of truth values
μ_f represents mean of model predictions
σ_y represents standard deviation of truth values
σ_f represents standard deviation of model predictions
E represents expectation

3 Model Architecture

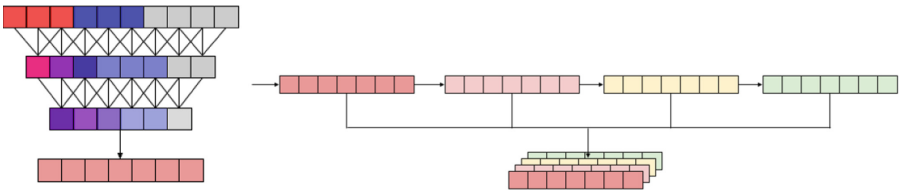
The proposed framework consists of architectures that comprise some or all of 3 main task-specific blocks namely Convolutional Block (CB), Recurrent Block (RB) and linear Block (LB).

3.1 Convolutional Block (CB)

This block makes use of 1 dimensional convolutional layers as mentioned in Sect. 2.2 to detect and extract characteristic patterns present in the data. Several works have leveraged the temporal application of convolutions for time series classification problems [7,8,12,13]. In time series forecasting, the kernels of the CNN, as mentioned in Sect. 2.2, convolve over the data along the temporal dimension, in other words, it explores how different patterns vary with time in the given data. Also instead of integrating, the results of the convolutions are stacked one behind the other as shown in Fig. 1c, giving multiple non-linear higher dimensional representations of the input data that can better disentangle the factors of variation among the input data [14]. The convolutions operate only along the temporal dimension of the data thus also maintaining its sequential context as shown in Fig. 3a. In essence, the convolutional layers work towards detecting short-term patterns present in the data and the use of stacked convolutions provide higher level representations of these patterns [15].

3.2 Recurrent Block (RB)

Recurrent Neural Networks (RNNs) are known for their wide ranges of applications in sequential learning tasks that spread across various fields such as natural language, speech recognition, audio and video processing [5,16]. Long Short-Term Memory Networks (LSTMs), which were originally described in [17], are a special kind of recurrent units capable of learning long-term dependencies present in the sequence data. The 3 gates (input, forget and output) along with the cell state enhances their capabilities in processing longer sequences when compared to simple RNN units. To further capture the temporal dependencies among the localized features generated by the CB, we introduce the Recurrent Block (RB) in the framework shown in Fig. 3b. However RNNs lack the capability to convert the raw input sequence to higher dimensions that can better distinguish the data points if used directly for sequence prediction [18]. While both CNN and the fully connected layers help combat the above mentioned issue, the latter can provide only 1 such representation whereas the former can provide multiple higher dimensional representation. Also, the localized features generated by CNN are temporally consistent giving the LSTMs a latent view of the original sequence. Hence the CB precedes the RB in the framework.



(a) Convolutional Block (b) Recurrent Block

Fig. 3. Block components

3.3 Linear Block (LB)

The last component is the Linear Block (LB). The outputs of the recurrent layers have a lot of variance among their hidden state values [18]. Also, the sequences and features from the RB and CB have to be merged to predict the target value in the current time frame. We make use of some fully connected layers in the LB to solve them. The fully connected layers learn to predict the target value using the information provided by both the CB and the RB through residual connections described in Sect. 3.4.

3.4 Residual Connections

Residual connections in the framework overcome vanishing gradients problem that might occur in the RB. Moreover, the CB learns directly from the errors of the LB, thus giving multiple learning sources during back propagation. The residual connections are added from the CB to the LB thus merging the features from the CB with the subsequences from the LB. These indirectly help in modeling the short-term and the long-term features of the data.

3.5 Model Variants

We set the following assumptions before generating the model variants.

- The training methodology of using future contextual frames is applied only to CB or RB.
- Variants with both CB and RB must have the CB preceding RB.
- Every variant comprising of CB has a residual link from CB to LB.
- Every variant must have either CB or RB before LB.

All architectural variants that abide by the above said assumptions were considered and are listed below.

Bi-CNN+LSTM+FC or M1. In this variant, there are two CBs, one RB and one LB. The abstract architectural design is shown in Fig. 4a. One of the CB takes only P data frames and the other takes only F' data frames. The output features from both the CBs are merged and sent into the RB. The RB produces a sequence using the combined information from both the CBs. We use residual connections from CB to the LB in this variant.

Bi-CNN+FC or M2. In this variant, there are two CBs and one LB. The abstract architecturd21 al design is shown in Fig. 4b. One of the CB operates on P data frames while the other operates on F' frames. The variant has no RB block so the features from both the CBs are merged and sent to the LB. We use this variant to check the importance of RB block in the architecture and also use it as a baseline model for time series forecast using CNNs.

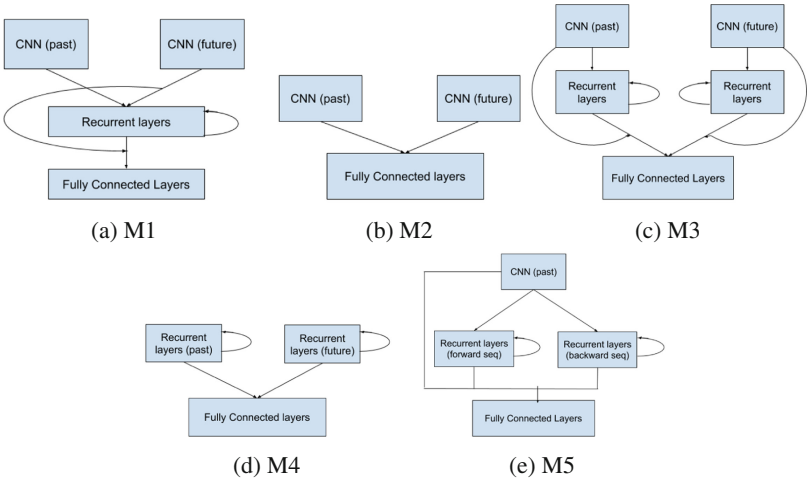


Fig. 4. Model variants

Bi-(CNN+LSTM)+FC or M3. In this variant, there are two CBs, two RBs and one LB. The abstract architectural design is shown in Fig. 4c. One of the CBs and one of the RBs are used to operate on the P data frames while the other CB and RB operate on F' data frames. The features from the respective CBs are fed into the respective RBs. The features and sequences generated by the combined CB+RB blocks are sent together to the LB. This variant is used to understand to which block should the future context be explicitly given and how to merge the block outputs.

Bi-LSTM or M4. In this variant, there are 2 RBs and one LB. The abstract architectural design is shown in Fig. 4d. One of the RBs takes P data frames as input while the other takes the F' data frames. The predicted sequences from both the RBs were merged and sent to the LB. We use this variant to understand the importance of CB in the architecture and to use it as a baseline model for time series forecast using RNNs.

CNN+Bi-LSTM+FC or M5. In this variant, there is one CB, two RBs and one LB. The abstract architectural design is shown in Fig. 4e. The CB operates only on P data frames. This variant is an adapted version of the CLDNN model [14] with bidirectional property added to RB. The feature outputs from CB are sent to one RB while the reversed outputs are sent to the other one. The combined sequences from both these RBs along with the features from CB are merged and sent to the LB.

4 Training and Testing Phase

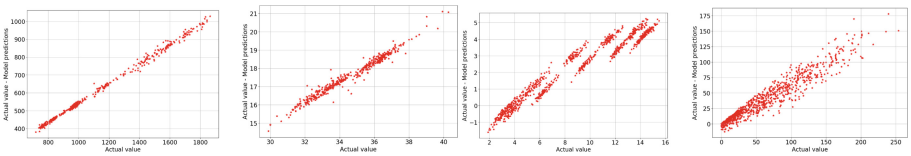
During the training phase, as shown in Fig. 2, the data is first segmented into 2 parts with respect to the current frame of reference. The convolutional branches extract features from P and F' data frames separately and pass it on to the next component of the model. The influence of the future frames are captured during training in the hidden states of the recurrent layers of RB and the weights of the fully connected layers of LB.

During the testing phase, only the P data frames are fed to the model to test its robustness in a simulated real-world scenario. Hence the F' data frames are set to 0, blacking out the entire future convolutional branch of the CB. To recapitulate, during the testing phase, the model relies only on the information provided by the past data frames.

The convolutional layer of the CB has 32 filters with kernel sizes of 5 and 2, along with a linear activation of the convolutions. The LSTM layer of the RB, has 32 hidden units with a *tanh* activation function. The dense layers of the LB have 256 neurons with a *LeakyRelu* activation. All the model variants are trained to minimize the *huber* loss function. The optimizers used are Stochastic Gradient Descent (SGD) and Adam, with custom learning rates.

5 Optimization

During testing, as the model is deprived of the future contextual frames, they do not contribute to the activations of the downstream neurons. Consequently, through the experiments conducted, it was observed that during the testing phase, the model predictions were always lower than the actual value despite not setting the weights of the LB to be positive. An optimizer is applied to offset the loss incurred during the testing phase, caused by blacking out the neurons corresponding to F' in the CB. In essence, an optimizer is to be used on top of all those model variants that use future context during the training phase. Figure 5 shows the scatter plot of the difference between model predictions and actual values against the actual values during the testing phase. From the plots it is evident that the model predictions and the actual values are linearly dependent for all domains of the data considered. Hence we use a linear regressor to optimize the model predictions.

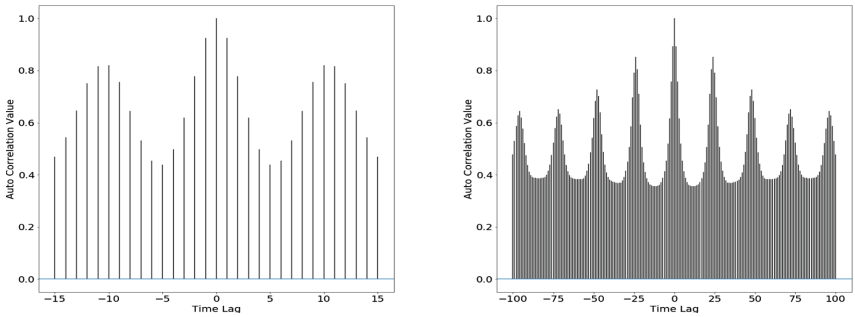


(a) Amazon Stock (b) Pfizer Stocks (c) Traffic Data (d) Wolfer Sunspot

Fig. 5. Linear difference of model predictions and actual values

6 Experimentation

To test the genericness and robustness of the framework, 3 specific classes of datasets were chosen that cut across diverse domains and have different statistical characteristics.



(a) Wolfer Sunspot (b) Traffic Dataset

Fig. 6. Autocorrelation plots

Wolf's Sunspot Dataset. The second dataset considered is the Wolf's Sunspot dataset, which is one of the well-known time series data sets. The data consists of sunspot numbers from 1700 to 1987 collected annually, giving a total of 288 observations. The data has an average seasonal cyclic pattern repeating itself over a mean period of about 10 years as observed from the autocorrelation plot in Fig. 6a. But experimentally it was observed that the framework performed best with k (refer Sect. 2.3) set as 7. The data is non-linear and non-Gaussian in nature that makes it harder to model than conventional time series datasets. The benchmark uses a hybrid of conventional ARIMA and ANN models [19] resulting in MSE of 186.27. From Table 1, it can be observed that M5 outperforms the baseline and also all the other variants.

Traffic Dataset. The first datatset is traffic data that contains the hourly road occupancy rates (between 0 and 1) measured over 185 sensors on San Francisco Bay area freeways during the years 2015–2016, provided by the California Department of Transportation. The autocorrelation plot in Fig. 6b shows the existence of a strong correlation at a periodicity of 24 h. Hence, the models were trained with k (refer Sect. 2.3) set as 24. Metrics for comparison are Correlation and Root Relative Squared Error (refer Sect. 2.4). The benchmark results are an RSE of 0.47 and a CORR of 0.87, achieved by LSTNet [20]. LSTNet combines the strengths of convolutional and recurrent neural networks with an autoregressive component. Results for 3 h ahead predictions are shown in Table 2. From Table 2 we can observe that M1 and M5 outperform the benchmark models by a great margin.

Table 1. Wolf's Sunspot results

Model variants	R2	RMSE
M1	0.855	18.32
M2	0.759	24.61
M3	0.707	27.12
M4	0.608	31.38
M5	**0.927**	**13.51**
G. Peter Zhang	Not available	13.66

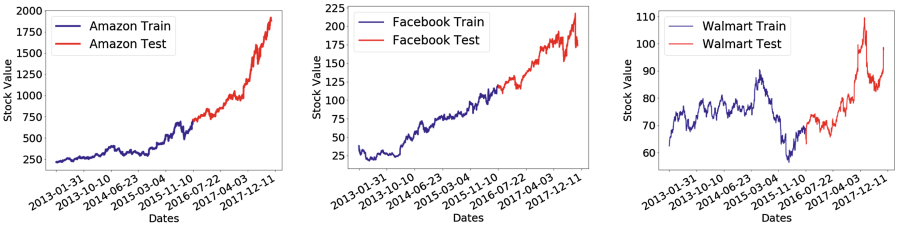
Table 2. Traffic dataset results

Model variants	RSE	CORR
M1	**0.4552**	**0.9242**
M2	0.5204	0.8795
M3	0.5450	0.8390
M4	0.5208	0.8539
M5	**0.4432**	**0.9724**
Guokun Lai et al.	0.4777	0.8721

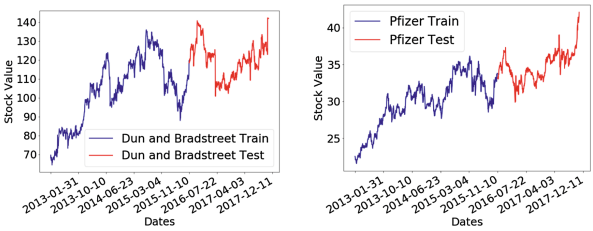
Stocks Dataset. The third class of datasets consists of daily closing stock prices of 5 different companies. It is known that stocks are event-based time series values that change erratically due to a plethora of factors that are hard to find and factor in. The uncertainty in the stock values arises due to a pool of external factors that could be recurrent or one-time events, thus giving stocks a non-seasonal cycle property sometimes, which can be seen in Fig. 7. To test the model's robustness, a few business-to-business (B2B) and a few business-to-consumer (B2C) company stocks were considered. It is known that B2B stocks depend on factors like inter-company relations, trust factors, etc. and are harder to predict than B2C stock values. The B2C stocks considered are Amazon, Walmart and Facebook. The B2B stocks considered are Dun & Bradstreet Corp and Pfizer Inc. The framework was trained across all stocks with k (refer Sect. 2.3) set as 10. The efficiencies of variants of framework have been tabulated below.

- It can be observed from Fig. 7a that the stock prices of Amazon have a global increasing trend. Steps, defined as a sudden permanent shift in values, exist throughout the data. The step values keep varying frequently, and hence there exists a difference between the current and future values within the contextual frames considered.
- In the case of Facebook stocks, Fig. 7b shows that the trend of the curve is progressive and smoother than Amazon. The step also changes frequently but there aren't any seasonal and non-seasonal cycles that might affect the performance of the model.
- In the case of Walmart stocks, Fig. 7c shows how significantly different it is from Facebook and Amazon, even though all these companies fall under the ambit of B2C. One reason could be because Walmart is a chain of retail stores, whereas the other two are technological ventures. The factors affecting them would be very different from the ones that influence the retail supermarket.
- In the case of Dun & Bradstreet and Pfizer stocks, there exists some seasonality which can be attributed to the fact that they are B2B. Compared to other stocks, these exhibit more irregularities which can be seen from Fig. 7d and e.

It can be observed from the Tables 3 and 4 that M1 performs the best across all the 5 stock datasets considered. Also, from these tables, we can observe that the framework performs better at univariate than multivariate forecasting.



(a) Amazon (b) Facebook (c) Walmart



(d) Dun & Bradstreet (e) Pfizer

Fig. 7. Stock curves

Table 3. Stocks dataset R^2 values

Model variants	PFE		WMT		AMZN		DNB		FB	
	Uni	Multi	Uni	Multi	Uni	Multi	Uni	Multi	Uni	Multi
M1	**0.988**	0.957	0.9878	0.983	**0.998**	0.995	**0.956**	0.943	**0.991**	0.971
M2	0.966	0.960	0.985	0.982	0.996	0.995	0.946	0.937	0.983	0.981
M3	0.965	0.949	0.987	0.986	0.994	0.993	0.947	0.936	0.984	0.983
M4	0.915	0.933	0.806	0.783	−4.35	−6.534	0.930	0.519	−2.07	−2.167
M5	0.963	0.961	**0.9882**	0.9881	0.996	0.996	0.952	0.948	0.982	0.982

Table 4. Stocks dataset $RMSE$ values

Model variants	PFE		WMT		AMZN		DNB		FB	
	Uni	Multi	Uni	Multi	Uni	Multi	Uni	Multi	Uni	Multi
M1	**0.20**	0.39	**0.72**	1.36	**8.22**	21.82	**1.09**	1.85	**1.37**	4.14
M2	0.34	0.37	0.82	1.40	13.67	23.58	1.18	1.94	1.86	3.36
M3	0.34	0.42	0.72	1.18	16.00	21.96	1.17	1.97	1.82	3.21
M4	0.54	0.48	2.20	4.80	68.53	78.34	1.35	5.39	37.24	43.39
M5	0.35	0.36	0.89	1.12	13.46	19.78	3.19	1.77	2.08	3.24

7 Conclusions

The proposed framework outperforms the baseline and the benchmark results across all the datasets considered. From Tables 3 and 4, we can observe that the framework performs better for univariate forecasting compared to

multivariate forecasting, on the stocks datasets. By comparing the results across these datasets, M5 performs better than M1 on datasets with seasonality and auto-correlative properties, however, M1 outperforms M5 as the complexity of the data increases. M1 and M2 evidently prove that the RB plays a vital role when it comes to conventional time series forecasting. The results of the variants M4 and M5 depict the importance of the CB in the proposed framework across all datasets. The point at which the data is merged (data-flow) plays a crucial role in the proposed framework which is brought out by the difference in the performance of M1 and M3 across the datasets.

In conclusion, this paper introduces a framework of architectures for time-series forecasting, consisting of three task-specific components: the CB, the RB and the LB. The framework is also trained using a novel training methodology, which involves the usage of future as well as past contextual frames.

References

1. Granger, C.W.J., Andersen, A.P.: An Introduction to Bilinear Time Series Models. Vandenhoeck & Ruprecht, Göttingen (1978)
2. Engle, R.F.: Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econom. J. Econom. Soc. **50**, 987–1007 (1982)
3. Tong, H.: Threshold Models in Non-linear Time Series Analysis, vol. 21. Springer, New York (2012)
4. De Gooijer, J.G., Kumar, K.: Some recent developments in non-linear time series modelling, testing, and forecasting. Int. J. Forecast. **8**(2), 135–156 (1992)
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
6. Bao, W., Yue, J., Rao, Y.: A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS One **12**(7), e0180944 (2017)
7. Bińkowski, M., Marti, G., Donnat, P.: Autoregressive convolutional neural networks for asynchronous time series (2017). arXiv preprint: arXiv:1703.04122
8. Borovykh, A., Bohte, S., Oosterlee, C.W.: Dilated convolutional neural networks for time series forecasting, March 2017
9. Cui, Z., Chen, W., Chen, Y.: Multi-scale convolutional neural networks for time series classification (2016). arXiv preprint: arXiv:1603.06995
10. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CoRR, abs/1411.4038 (2014)
11. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. Trans. Signal Process. **45**(11), 2673–2681 (1997)
12. Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L.: Time series classification using multi-channels deep convolutional neural networks. In: Li, F., Li, G., Hwang, S., Yao, B., Zhang, Z. (eds.) WAIM 2014. LNCS, vol. 8485, pp. 298–310. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08010-9_33
13. Yang, J., Nguyen, M.N., San, P.P., Li, X., Krishnaswamy, S.: Deep convolutional neural networks on multichannel time series for human activity recognition. In: IJCAI, vol. 15, pp. 3995–4001 (2015)

14. Sainath, T., Vinyals, O., Senior, A., Sak, H.: Convolutional, long short-term memory, fully connected deep neural networks. In: ICASSP (2015)
15. Lin, T., Guo, T., Aberer, K.: Hybrid neural networks for learning the trend in time series. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 2273–2279 (2017)
16. Graves, A., Mohamed, A.-R., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6645–6649. IEEE (2013)
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
18. Pascanu, R., Gulcehre, C., Cho, K., Bengio, Y.: How to construct deep recurrent neural networks. In: Proceedings of the Second International Conference on Learning Representations (ICLR 2014) (2014)
19. Peter Zhang, G.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing **50**, 159–175 (2003)
20. Lai, G., Chang, W.-C., Yang, Y., Liu, H.: Modeling long- and short-term temporal patterns with deep neural networks. CoRR, abs/1703.07015 (2017)