An Analysis of Deep Neural Networks for Predicting Trends in Time Series Data

Kouame Hermann Kouassi[1,2(✉)] and Deshendran Moodley[1,2]

[1] University of Cape Town, 18 University Avenue Rondebosch, Cape Town 7700, South Africa
ksskou001@myuct.ac.za, deshen@cs.uct.ac.za

[2] Centre for Artificial Intelligence Research, 18 University Avenue Rondebosch, Cape Town 7700, South Africa

Abstract. Recently, a hybrid Deep Neural Network (DNN) algorithm, TreNet was proposed for predicting trends in time series data. While TreNet was shown to have superior performance for trend prediction to other DNN and traditional ML approaches, the validation method used did not take into account the sequential nature of time series datasets and did not deal with model update. In this research we replicated the TreNet experiments on the same datasets using a walk-forward validation method and tested our best model over multiple independent runs to evaluate model stability. We compared the performance of the hybrid TreNet algorithm, on four datasets to vanilla DNN algorithms that take in point data, and also to traditional ML algorithms. We found that in general TreNet still performs better than the vanilla DNN models, but not on all datasets as reported in the original TreNet study. This study highlights the importance of using an appropriate validation method and evaluating model stability for evaluating and developing machine learning models for trend prediction in time series data.

Keywords: Time series trend prediction · Deep neural networks · Ensemble methods · Walk-forward validation

1 Introduction

With the advent of low cost sensors and digital transformation, time series data is being generated at an unprecedented speed and volume in a wide range of applications in almost every domain. For example, stock market fluctuations, computer cluster traces, medical and biological experimental observations, sensor networks readings, etc., are all represented in time series. Consequently, there is an enormous interest in analyzing time series data, which has resulted in a large number of studies on new methodologies for indexing, classifying, clustering, summarizing, and predicting time series data [10,11,16,23,25].

Centre for Artificial Intelligence Research (CAIR).

A. Gerber (Ed.): SACAIR 2020, CCIS 1342, pp. 119–140, 2020.
https://doi.org/10.1007/978-3-030-66151-9_8

In certain time series prediction applications, segmenting the time series into a sequence of trends and predicting the slope and duration of the next trend is preferred over predicting just the next value in the series [16,23]. Piecewise linear representation [10] or trend lines can provide a better representation for the underlying semantics and dynamics of the generating process of a non-stationary and dynamic time series [16,23]. Moreover, trend lines are a more natural representation for predicting change points in the data, which may be more interesting to decision makings. For example, suppose a share price in the stock market is currently rising. A trader in the stock market would ask "How long will it take and at what price will the share price peak and when will the price start dropping?" Another example application is for predicting daily household electricity consumption. Here the user may be more interested in identifying the time, scale and duration of peak or low energy consumption.

While deep neural networks (DNNs) has been widely applied to computer vision, natural language processing (NLP) and speech recognition, there is limited research on applying DNNs for time series prediction. In 2017, Lin et al. [16] proposed a novel approach to directly predict the next trend of a time series as a piecewise linear approximation (*trend line*) with a slope and a duration using a hybrid neural network approach, called TreNet. The authors showed that TreNet outperformed SVR, CNN, LSTM, pHHM [23], and cascaded CNN and RNN.

However, the study had certain limitations.

Inadequacy of Cross-validation: The study used standard cross-validation with random shuffling. This implies that data instances, which are generated after a given validation set, are used for training [1].

No Model Update: In real world applications where systems are often dynamic, models become outdated and must be updated as new data becomes available. TreNet's test error was estimated on a single hold-out set, which assumes that the system under consideration is static. TreNet's evaluation therefore does not provide a sufficiently robust performance measure for datasets that are erratic and non-stationary [1].

No Evaluation of Model Stability: DNNs, as a result of random initialisation and possibly other random parameter settings could yield substantially different results when re-run with the same hyperparameter values on the same dataset. Thus, it is crucial that the best DNN configurations should be stable, i.e. have minimal deviation from the mean test loss across multiple runs. There is no evidence that this was done for TreNet.

Missing Implementation Details: Important implementation details in the TreNet study are not stated explicitly. For instance, the segmentation method used to transform the raw time series into trend lines is not apparent. This questions the reproducibility of TreNet's study.

This paper attempts to address these shortcomings. Our research questions are:

1. Does a hybrid deep neural networks approach for trend prediction perform better than vanilla deep neural networks?

2. Do deep neural networks models perform better for trend prediction than simpler traditional machine learning (ML) models?
3. Does the addition of trend line features improve performance over local raw data features alone?

The remainder of the paper is structured as follows. We first provide a brief background of the problem and a summary of related work, followed by the experimental design. We then give a brief overview of the experiments, describe the experiments, present and discuss their results. Finally, we provide a summary and discussion of the key findings.

2 Background and Related Work

2.1 Background

The time series trend prediction problem is concerned with predicting the future evolution of the time series from the current time. This evolution is approximated as a succession of time-ordered piecewise linear approximations. The linear approximations indicate the *direction*, the *strength*, and the *length* of the upward/downward movement of the time series. The *slope* of the linear approximation determines the *direction* and the *strength* of the movement, and the number of time steps covered by that linear approximation, i.e. its *duration* determines its *length*. The formal problem definition is given below.

Problem Formulation: We define a univariate time series as $X = \{x_1, ..., x_T\}$, where x_t is a real-valued observation at time t. The trend sequence T for X, is denoted by $T = \{<l_1, s_1>, ..., <s_k, l_k>\}$, and is obtained by performing a piecewise linear approximation of X [10]. l_k represents the *duration* and is given by the number of data points covered by trend k and s_k is the slope of the trend expressed as an angle between –90 and 90°. Given a historical time series X and its corresponding trend sequence T, the aim is to predict the *next trend* $<s_{k+1}, l_{k+1}>$.

2.2 Related Work

Traditional trend prediction approaches include Hidden Markov Models (HMM)s [19,23] and multi-step ahead predictions [2]. Leveraging the success of CNNs, and LSTMs in computer vision and natural language processing [3,7,14], Lin et al. [16] proposed a hybrid DNN approach, TreNet, for trend prediction. TreNet uses a CNN which takes in recent point data, and an LSTM which takes in historical trend lines to extract local and global features respectively. These features are then fused to predict the next trend. While the authors report a marked performance improvement when compared to other approaches, the validation method used in their experiments is questionable. More specifically it does not take into account the sequential nature of times series data. The data was first randomly shuffled, 10% of the data was held out for testing and a cross validation approach for training with the remainder of the data. Randomly shuffling the data

and using a standard cross validation approach does not take into account the sequential nature of time series data and may give erroneous results [16]. A walk-forward validation with successive and overlapping partitioning (see Sect. 3.4) is better suited for evaluating and comparing model performance on time series data [18]. It maintains the order of a time series sequence and deals with changes in its properties over time [18]. To deal with this limitation we attempt to replicate the TreNet approach using a walk forward validation instead of random shuffling and cross validation.

Some follow-up research to TreNet added attention mechanisms [5,27], however, they not deal with trend prediction specifically. Another active and related field to trend prediction is the stock market direction movement, which is only concerned with the direction of the time series, it does not predict the strength and the duration of the time series [5,6,9,17,20,24]. Generally, the baseline methods used by prior work include neural networks, the naive last value prediction, ARIMA, SVR [16,26]. They do not include ensemble methods such as random forests, which are widely used particularly for stock market movement prediction [13,22].

3 Experimental Design

3.1 Datasets

Experiments were conducted on the four different datasets described below.

1. The *voltage dataset* from the UCI machine learning repository[1]. It contains 2075259 data points of a household voltage measurements of one minute interval. It is highly volatile but normally distributed. It follows the same pattern every year, according to the weather seasons as shown in Fig. 4 in the appendix. It corresponds to the power consumption dataset used by Lin et al. [16].
2. The *methane dataset* from the UCI machine learning repository[2]. We used a resampled set of size of 41786 at a frequency 1 Hz. The methane dataset is skewed to the right of its mean value and exhibits very sharp changes with medium to low volatility as shown in Fig. 5 in the appendix. It corresponds to the gas sensor dataset used by Lin et al. [16].
3. The *NYSE dataset* from Yahoo finance[3]. It contains 13563 data points of the composite New York Stock Exchange (NYSE) closing price from 31-12-1965 to 15-11-2019. Its volatility is very low initially until before the year 2000 after which, it becomes very volatile. It is skewed to the right as shown in Fig. 6 in the appendix. It corresponds to the stock market dataset used by Lin et al. [16].

[1] https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption.

[2] https://archive.ics.uci.edu/ml/datasets/gas+sensor+array+under+dynamic+gas+mixtures.

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

4. The *JSE dataset* from Yahoo finance. It contains 3094 data points of the composite Johannesburg Stock Exchange (JSE) closing price from 2007-09-18 to 2019-12-31. Compared to the NYSE, this stock market dataset is less volatile and shows a symmetrical distribution around its mean value. However, it has a flat top and heavy tails on both sides as shown in Fig. 7 in the appendix.

The characteristics of the four datasets are summarised in Table 1.

Table 1. Summary of the characteristics of the datasets.

	Seasonality	Skewness	Volatility
Voltage	Seasonal	Symmetric	Very high
Methane	Non-seasonal	Right skewness	Medium to low
NYSE	Non-seasonal	Right skewness	Low to high
JSE	Non-seasonal	Almost symmetric	Medium to low

3.2 Data Preprocessing

The data preprocessing consists of three operations: missing data imputation, the data segmentation, and the sliding window operation. Each missing data point is replaced with the closest preceding non-missing value. The segmentation of the time series into trend lines i.e. piecewise linear approximations is done by regression using the bottom-up approach, similar to the approach used by Wang et al. [23]. The data instances, i.e. the input-output pairs are formed using a sliding window. The input features are the local data points $L_k = <x_{t_k-w}, ..., x_{t_k}>$ for the current trend $T_k = <s_k, l_k>$ at the current time t. The window size w is determined by the duration of the first trend line. The output is the next trend $T_{k+1} = <s_{k+1}, l_{k+1}>$. The statistics of the segmented datasets are provided in Table 7 in the appendix.

3.3 Learning Algorithms

The performance of seven ML algorithms, i.e. the hybrid TreNet approach, four vanilla DNN algorithms and two traditional ML algorithms were evaluated. These algorithms are described below.

TreNet: TreNet has a hybrid CNN which takes in raw point data, and LSTM which takes in trend lines as shown in Fig. 1. The LSTM consisted of a single LSTM layer, and the CNN is composed of two stacked [16] 1D convolutional neural networks without pooling layer. The second CNN layer is followed by a ReLU activation function. Each of the flattened output of the CNN's ReLU layer and the LSTM layer is projected to the same dimension using a fully connected

layer for the fusion operation. The fusion layer consists of a fully connected layer that takes the element-wise addition of the projected outputs of the CNN and LSTM components as its input, and outputs the slope and duration values. A dropout layer is added to the layer before the output layer. The best TreNet hyperparameters for each dataset are shown in Table 9 in the appendix and compared to Lin et al.'s [16].

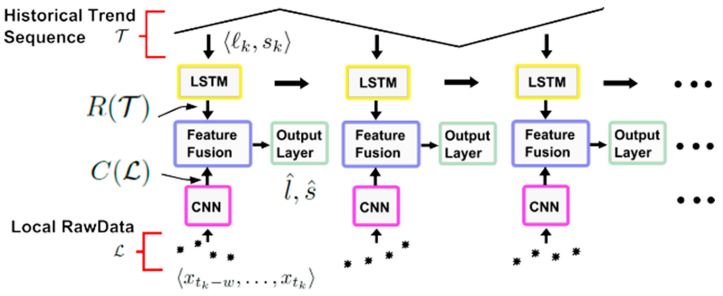

Fig. 1. Illustration of the hybrid neural network architecture [16]

Vanilla DNN Algorithms:

- *The MLP* consists of N number of fully connected neural network (NN) layers, where, $N \in [1, 5]$. Each layer is followed by a ReLU activation function to capture non-linear patterns. To prevent overfitting, a dropout layer is added after each odd number layer, except the last layer. For instance, if the number of layers $N = 5$, the layer 1 and layer 3 will be followed by a dropout layer.
- *The LSTM* consists of N LSTM layers, where $N \in [1, 3]$. Each layer is followed by a ReLU activation function to extract non-linear patterns, and a dropout layer to prevent overfitting. After the last dropout layer, a fully connected NN layer is added. This layer takes the feature representation extracted by the LSTM layers as its input and predicts the next trend. The LSTM layers are not re-initialised after every epoch.
- *The CNN* consists of N 1D-convolutional layer, where $N \in [1, 3]$. Each convolutional layer, which consists of a specified number of filters of a given kernel size, is followed by a ReLU activation function, a pooling layer, and a dropout layer to prevent overfitting. The final layer of the CNN algorithm is a fully connected neural network which takes the features extracted by the convolution, activation, pooling, and dropout operations as its input and predicts the next trend. The structure of a 1D-CNN layer is illustrated in Fig. 2.

The parameters of the vanilla DNN algorithms were tuned manually. The best values found for each algorithm on each dataset are shown in Table 10 in the appendix.

DNN Algorithm Training, and Initialisation: The equally weighted average slope and duration mean square error (MSE) is used as a loss function during training with the Adam optimizer [12]. To ensure robustness against random initialisation, the DNNs are initialised using the He initialisation technique [8] with normal distribution, fan-in mode, and a ReLU activation function.

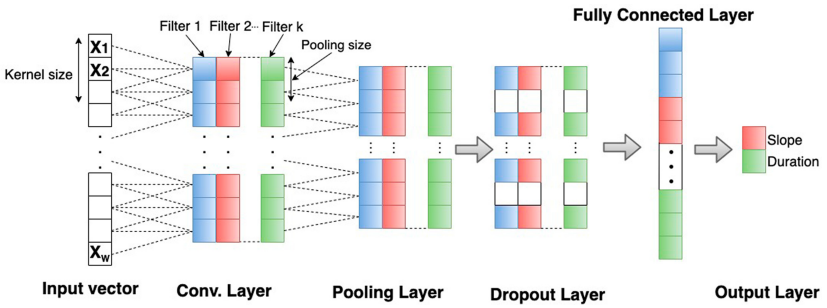

Fig. 2. The structure of a one layer 1D-convolution neural network.

Traditional ML Algorithms: The SVR and RF algorithms are implemented using Sklearn [21], but, GBM is implemented with LightGBM[4]. We tuned the *gamma* and *C* hyperparameters for the SVR; the *number of estimators*, the *maximum depth*, the *boostrap*, and *warm start* hyperparameters for the RF; as well as the *boostrap type*, the *number of estimators*, and the *learning rate* hyperparameters for the GBM, on the validation sets. Their best hyperparameter configurations per dataset are shown in Table 11 in the appendix. We use the MSE loss for the RF.

3.4 Model Evaluation with Walk-Forward Evaluation

The walk-forward evaluation procedure, with the successive and overlapping training-validation-test partition [18], is used to evaluate the performance of the models. The input-output data instances are partitioned into training, validation, and test sets in a successive and overlapping fashion [18] as shown in Fig. 3. For the methane and JSE datasets, the combined test sets make up 10% of their total data instances as per the original TreNet experiments; and 80% and 50% for the voltage and NYSE datasets respectively because of their large sizes. The partition sizes for each dataset are given in Table 8 in the appendix. We set the number of partitions to 8 for the voltage, 44 for methane, 5 for NYSE and, 101 for the JSE dataset. This determines the number of model updates performed for each dataset. For example, one initial training and 7 (8-1) model updates are performed for the voltage dataset. For DNN models, the neural networks

[4] https://lightgbm.readthedocs.io/en/latest/index.html.

are initialised using the weights of the most recent model, during model update. This makes the training of the network faster without compromising its generalisation ability. More details about this technique which we refer to as *model update with warm-start* is given in Sect. A.2 in the appendix. The average root mean square error (RMSE), given in Eq. 1, is used as the evaluation metric.

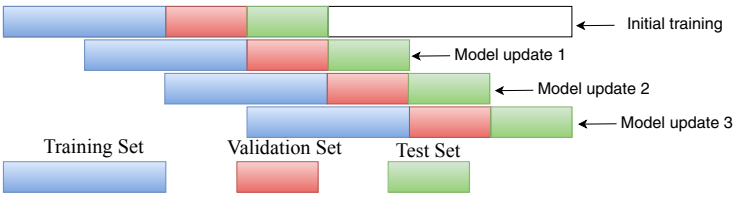

Fig. 3. An example of successive and overlapping training-validation-test partitioning with 4 partitions (1 initial training and 3 model updates)

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - y_t')^2} \tag{1}$$

where, $y_t \rightarrow$ actual next trend, $y_t^{'} \rightarrow$ predicted next trend, and $T \rightarrow$ number of data instances. For the DNN algorithms each experiment is run 10 times and the mean and the standard deviation across the 10 runs are reported. This provides a measure of the stability of the DNN configuration using different random seeds.

4 Experiments

We performed four experiments; each with four datasets. In experiment 1, we implement and evaluate a TreNet [16]. TreNet uses a hybrid deep learning structure, that combines both a CNN and an LSTM, and takes in a combination of raw data points and trend lines as its input. In experiment 2, we compared the TreNet results with the performance of vanilla MLP, CNN and LSTM structures on raw point data to analyse the performance improvement when using a hybrid approach with trend lines. In experiment 3, we evaluate the performance of three traditional ML techniques, i.e. SVR, RF, and GBM on raw point data to analyse the performance difference between DNN and non-DNN approaches. In experiment 4, we supplement the raw data features with trend lines features to evaluate the performance improvement over the raw data features alone for both DNN and non-DNN algorithms.

4.1 Experiment 1: Replicating TreNet with Walk-Forward Validation

We replicated the TreNet approach using a walk forward validation rather than random shuffling and cross validation used in the original TreNet experiments.

Table 2. Comparison of the slope (S), duration (D), and average (A) RMSE values achieved by our hybrid neural network's performance and Lin et al.'s results. The percentage improvement (% improv.) over the naive LVM

	Voltage			Methane			NYSE		
	S	D	A	S	D	A	S	D	A
Our LVM	17.09	86.51	51.80	28.54	152.86	90.70	127.16	0.33	63.75
Our TreNet	9.25	62.37	35.81	14.87	31.25	23.06	86.89	1.23	44.06
Our % improv.	**45.87**	27.90	30.87	**47.90**	**79.56**	**74.58**	**31.67**	−272.73	**30.89**
Lin et al.'s LVM	21.17	39.68	30.43	10.57	53.76	32.17	8.58	11.36	9.97
Lin et al.'s TreNet	12.89	25.62	19.26	9.46	51.25	30.36	6.58	8.51	7.55
Lin et al.'s % improv.	39.11	**35.43**	**36.71**	10.50	4.69	5.63	23.31	**25.09**	24.27

In order to compare our results with the original TreNet we use a similar performance measure to Lin et al. [16]. We measure the percentage improvement over a naive last value model (LVM). The naive last value model simply "takes the duration and slope of the last trend as the prediction for the next one" [16]. The use of a relative metric makes comparison easier, since the RMSE is scale-dependent, and the trend lines generated in this study may differ from Lin et al.'s [16]. Lin et al. [16] did not provide details of the segmentation method they used in their paper. Furthermore, the naive last value model does not require any hyper-parameter tuning, its predictions are stable and repeatable, i.e. does not differ when the experiment is rerun, and is only dependent on the characteristics of the dataset.

Table 2 shows the performance improvement on RMSE values over the LVM achieved by the TreNet implementation on each dataset. They are compared to the performance of the original TreNet on the three datasets they used in their experiments, i.e. the voltage, methane and NYSE datasets. The results of our experiment differ substantially from those reported for the original TreNet. Our TreNet models' percentage improvement over the naive LVM is 13.25 (**74.58**/5.63) and 1.27 (**30.89**/24.27) times greater than Lin et al.'s [16], on the methane and NYSE datasets respectively; but 1.19 (36.71/**27.90**) times smaller on the voltage dataset. The naive LVM performs better than our TreNet model on the NYSE for the duration prediction. The –272.73 % decrease in performance is due to two reasons. On the one hand, the model training, i.e. the loss minimisation was biased towards the slope loss at the expense of the duration loss. This is because the slope loss significantly greater compared to the duration loss, but, TreNet's loss function weights both equally. On the other hand, the durations of the trends in the NYSE dataset being very similar - with a standard deviation of 0.81 - makes the last value prediction model a favourably competitive model for the duration prediction.

The greater average improvement on the methane and NYSE is attributed to the use of the walk-forward evaluation procedure. The methane and NYSE datasets undergo various changes in the generating process because of the sudden changes in methane concentrations and the economic cycles for the NYSE. Thus,

the use of the walk-forward evaluation ensures that the most recent and useful training set is used for a given validation/test set. However, given that Lin et al. [16] did not drop older data from the training data set, the network may learn long-range relationships that are not useful for the current test set. Furthermore, they used random shuffling which may most likely result in future data points being included in the training data. The smaller improvement of our TreNet model on the voltage dataset can be attributed to our use of a smaller window size for the *local raw data* fed into the CNN. We used 19 compared to their best value of 700 on the voltage dataset. This is one of the limitations of our replication of TreNet. For each dataset, we used the length of the first trend line as window size of the *local raw data* feature fed into the CNN, instead of tuning it to select the best value. The other limitation is the use of a sampled version of the methane dataset instead of the complete methane dataset.

4.2 Experiment 2: Trend Prediction with Vanilla DNN Algorithms

Given that we are now using a different validation method which yields different performances scores to the original TreNet, we checked whether the TreNet approach still outperforms the vanilla DNN algorithms. We implemented and tested three vanilla DNN models namely a MLP, LSTM, and CNN using only raw local data features.

Table 3 shows the average RMSE values for slope and trend predictions achieved by the vanilla DNNs and TreNet on each dataset across 10 independent runs. The deviation across the 10 runs is also shown to provide an indication of the stability of the model across the runs. We use the average slope and duration RMSE values as an overall comparison metric. The % improvement is the improvement of the best vanilla DNN model over TreNet. The best model is chosen based on the overall comparison metric.

In general TreNet still performs better than the vanilla DNN models, but does not outperform the vanilla models on all the datasets. The most noticeable case is on the NYSE, where the LSTM model outperforms the TreNet model on both the slope and duration prediction. This contradicts Lin et al. [16]'s findings, where TreNet clearly outperforms all other models including LSTM. On average, Lin et al.'s [16] TreNet model outperformed their LSTM model by 22.48%; whereas, our TreNet implementation underperformed our LSTM model by 1.31%. However, Lin et al. [16]'s LSTM model appears to be trained using trend lines only and not raw point data. This LSTM model uses *local raw data* features. It must also be noted that the validation method used here is substantially different from the one used by Lin et al. [16]. The large performance difference between TreNet and the vanilla models on the methane dataset is because for this dataset the raw local data features do not provide the global information about the time series since it is non-stationary. This is confirmed by the increase in the performance of the MLP (23.83%), LSTM (11.02%) and CNN (24.05%) after supplementing the raw data features with trend line features (see experiment 4 in Sect. 4.4).

Table 3. Comparison of the RMSE values achieved by the vanilla DNN models and TreNet. The % improvement (% improv.) is the improvement of the best vanilla DNN model over TreNet

	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
MLP	**9.04 ± 0.06**	62.82 ± 0.04	35.93 ± 0.05	14.57 ± 0.10	49.79 ± 4.85	32.18 ± 2.48
LSTM	10.30 ± 0.0	62.87 ± 0.0	36.59 ± 0.0	**14.21 ± 0.19**	56.37 ± 1.77	35.29 ± 0.49
CNN	9.24 ± 0.10	62.40 ± 0.13	35.82 ± 0.12	15.07 ± 0.35	54.79 ± 4.55	34.93 ± 2.45
TreNet	9.25 ± 0.0	**62.37 ± 0.01**	**35.81 ± 0.01**	14.87 ± 0.40	**31.25 ± 2.62**	**23.06 ± 1.51**
% improv.	−0.11	−0.05	−0.03	**2.02**	−59.33	−39.55
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
MLP	90.76 ± 4.43	33.08 ± 42.08	61.92 ± 23.26	19.87 ± 0.01	12.51 ± 0.09	16.19 ± 0.05
LSTM	**86.56 ± 0.01**	**0.41 ± 0.08**	**43.49 ± 0.05**	19.83 ± 0.01	12.68 ± 0.01	16.25 ± 0.01
CNN	89.31 ± 1.38	12.21 ± 12.17	50.76 ± 6.78	19.90 ± 0.06	**12.48 ± 0.21**	16.19 ± 0.14
TreNet	86.89 ± 0.14	1.23 ± 0.38	44.06 ± 0.26	**19.65 ± 0.05**	12.49 ± 0.04	**16.07 ± 0.05**
% improv.	**0.38**	**66.67**	**1.29**	−1.12	−0.16	−0.75

4.3 Experiment 3: Traditional ML Models

Given the new validation method, we now compare the performance of DNN trend prediction models to the performance of traditional ML models. We implemented and tested three traditional ML models, i.e. radial-based SVR, RF, and GBM. To our knowledge, RF and GBM have not been used previously for trend prediction. Lin et al. [16] compared their approach against multiple SVR kernels that took in both local raw data and trend line features. In this experiment, our models take in only *local raw data* features without trend lines.

Table 4 shows the RMSE values achieved by the traditional ML algorithms and the best DNN models on each dataset. The best DNN model is TreNet on all datasets except on the NYSE, on which LSTM is the best model. The improvement (%) is the performance improvement of the best traditional ML model over the best DNN model, where, the best model is selected based on the equally weighted average slope and duration RMSE, i.e. average.

The best traditional ML algorithm underperformed the best DNN algorithm by 0.47% and 1.74% respectively on the (almost) normally distributed datasets such voltage and the JSE datasets. However, the RF model outperformed the best DNN model, i.e. TreNet by 33.04% on the methane dataset; while the SVR model matched the performance of the best DNN model, i.e. LSTM on the NYSE dataset. TreNet learns long-range dependencies from trend line features with its LSTM component. Although this is useful for stationary and less evolving time series such as the voltage and JSE datasets, it appears that it can be detrimental in the case of dynamic and non-stationary time series such as the methane dataset. This may explain why the traditional ML models, which do not keep long-term memory, performed better on this dataset.

Table 4. Comparison of the best DNN models (Best DNN) with the traditional ML algorithms. The % improvement (% improv.) is the performance improvement of the best traditional ML model over the best DNN model

	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
RF	9.53 ± 0.0	63.11 ± 0.20	36.32 ± 0.10	**10.09 ± 0.01**	**20.79 ± 0.01**	**15.44 ± 0.01**
GBM	10.0 ± 0.0	62.67 ± 0.0	36.34 ± 0.0	13.05 ± 0.0	75.10 ± 0.0	44.08 ± 0.0
SVR	9.32 ± 0.0	62.58 ± 0.0	35.95 ± 0.0	14.98 ± 0.0	34.39 ± 0.0	24.69 ± 0.0
Best DNN	**9.25 ± 0.0**	**62.37 ± 0.01**	**35.81 ± 0.01**	14.87 ± 0.40	31.25 ± 2.62	23.06 ± 1.51
% improv.	−0.76	−0.34	−0.47	**32.15**	**33.47**	**33.04**
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
RF	88.75 ± 0.17	**0.29 ± 0.0**	44.52 ± 0.09	20.21 ± 0.0	12.67 ± 0.0	16.44 ± 0.0
GBM	86.62 ± 0.0	0.42 ± 0.0	43.52 ± 0.0	20.08 ± 0.0	12.62 ± 0.0	16.35 ± 0.0
SVR	**86.55 ± 0.0**	0.42 ± 0.0	43.49 ± 0.0	20.01 ± 0.0	12.85 ± 0.0	16.43 ± 0.0
Best DNN	86.56 ± 0.01	**0.41 ± 0.08**	43.49 ± 0.05	**19.65 ± 0.05**	**12.49 ± 0.04**	**16.07 ± 0.05**
% improv.	**0.01**	2.44	0.0	−2.19	−1.04	−1.74

The fact that the radial-based SVR performed better than TreNet on the NYSE dataset contradicts Lin et al. [16]'s results. We attribute this to the use of *local raw data* features alone, instead of *local raw data* plus *trend line* features used by Lin et al. [16].

4.4 Experiment 4: Addition of Trend Line Features

In this experiment, we supplement the raw data with trend line features to analyse whether this yields any performance improvement to the DNN and non-DNN models from Experiments 2 and 3. We did retain the hyperparameter values found using the *raw data* features alone for this experiment.

Table 5 shows the average performance improvement (%) after supplementing the raw data with trend line features. The negative sign indicates a drop in performance. The Average is the mean and the standard error of the improvements over the algorithm or the dataset. The actual RMSE values are shown in Table 12 and Table 13 in the appendix.

Table 5. Performance improvement after supplementing the raw data with trend line features.

	MLP	LSTM	CNN	RF	GBM	SVR	Average
Voltage	**0.03**	0.0	−73.14	−0.13	**0.06**	−0.36	−12.26 ± 12.18
Methane	**23.83**	**11.02**	**24.05**	−4.47	**42.88**	−6.28	**15.17 ± 7.71**
NYSE	**6.49**	0.0	−1.00	**2.36**	**0.23**	−0.02	**1.34 ± 1.12**
JSE	−4.14	−1.17	−5.37	−7.60	**0.37**	−10.96	−4.81 ± 1.70
Average	**6.55 ± 6.16**	**4.93 ± 2.87**	−13.87 ± 20.79	−2.46 ± 2.22	**10.89 ± 10.67**	−4.41 ± 2.62	

The addition of trend line features improved the performance of both DNN and non-DNN models 10 times out of 24 cases. In general, it improves the performance of dynamic and non-stationary time series such as the methane and NYSE datasets. This is because local raw data features do not capture the global information about the time series for non-stationary time series. Thus, the addition of trend line features brings new information to the models. In 12 out of 24 cases, the addition of trend line features reduced the performance of both DNN and non-DNN models except the GBM models. For these cases, the addition trend line features brings noise or duplicate information, which the models did not deal with successfully. This may be because the best hyperparameters for the raw data features alone may not be optimal for the raw data and the trend line features combined. For instance, DNN models are generally able to extract the true signal from noisy or duplicate input features, however, they are sensitive to the hyperparameter values.

The above results show that the addition of trend line features has the potential to improve the performance of both DNN and non-DNN models on non-stationary time series. This comes at the cost of additional complexities and restrictions. The first complexity is related to the model complexity because the bigger the input feature size, the more complex the model becomes. Secondly, the trend line features require the segmentation of the time series into trends, which brings new challenges and restrictions during inference. For instance, trend prediction applications that require online inference need an online segmentation method such as the one proposed by Keogh et al. [10]. It is therefore necessary to evaluate whether the performance gain over raw data features alone justifies these complexities and restrictions.

4.5 Summary of the Best Performing Trend Prediction Algorithms

Table 6 provides a summary of the best models and their average performance from all four experiments. The TreNet algorithm outperforms the non-hybrid algorithms on the voltage and JSE datasets, but the performance difference is marginal $< 1\%$. Interestingly, the traditional ML algorithms outperformed TreNet and the vanilla DNN algorithms on the methane and NYSE datasets.

The additional of trend lines to the point data (experiment 4) did not yield any substantial change in the results. It must be noted though that this was an exploratory experiment and that no hyper-parameter optimisation was done to cater for the introduction of a new input feature. It may well be the case that better models could be found of a new hyper-parameter optimisation process was undertaken.

It is clear from these results that TreNet generally performs well on most datasets. However, it is not the clear winner, and there are some dataset where traditional models can substantially outperform TreNet. It is also clear that models built with point data alone can generally reach the performance levels of TreNet.

Table 6. Average RMSE values (E) achieved by the hybrid algorithm, i.e. TreNet; and the best non-hybrid algorithm (A) with raw point data features alone (Pt) and with raw point data plus trend line features (Pt + Tr). The % change is with respect to the TreNet algorithm.

		% Change	Pt	Hybrid	Pt + T	% Change
Voltage	*A*	–	CNN	**TreNet**	MLP	–
	E	−0.03	35.82 ± 0.12	$\mathbf{35.81 \pm 0.01}$	35.92 ± 0.05	−0.31
Methane	*A*	–	**RF**	TreNet	RF	–
	E	**33.04**	$\mathbf{15.44 \pm 0.01}$	23.06 ± 1.51	16.13 ± 0.01	30.05
NYSE	*A*	–	SVR	TreNet	**GBM**	–
	E	**1.29**	43.49 ± 0.0	44.06 ± 0.26	$\mathbf{43.42 \pm 0.0}$	1.45
JSE	*A*	–	MLP	**TreNet**	GBM	–
	E	−0.75	16.19 ± 0.05	$\mathbf{16.07 \pm 0.05}$	16.29 ± 0.0	−1.37

5 Discussion and Conclusions

In this work, we identify and address some limitations of a recent hybrid CNN and LSTM approach for trend prediction, i.e. TreNet. We used an appropriate validation method, i.e. walk-forward validation instead of the standard cross-validation and also tested model stability. We compared TreNet to vanilla deep neural networks (DNNs) that take in point data features. Our results show that TreNet does not always outperform vanilla DNN models and when it does, the outperformance is marginal. Furthermore, our results show that for non-normally distributed datasets, traditional ML algorithms, such as Random Forests and Support Vector Regressors, can outperform more complex DNN algorithms. We highlighted the importance of using an appropriate validation strategy and testing the stability of DNN models when they are updated and retrained as new observations become available.

There are many avenues to probe the results of this work further. Firstly we only tested this on four datasets. While these included all three datasets used in the original TreNet paper [16], testing on more datasets is required to probe the generalisation of these findings. Secondly, there are some avenues that can be explored to improve on these results. Since the window size was fixed to the duration length of the first trend line, the effect of varying the window size could be tested. A sampled version of the methane dataset is used instead of the complete methane dataset and the full dataset could be used. Tuning the hyperparameters of the model after the addition of trend line features may increase the performance of the models. Finding the best hyper-parameter values for a particular time series required extensive experimentation, and often requires information about the characteristics of that time series. Automatic machine learning techniques [4,15] could be explored for automating the feature selection, algorithm selection and hyperparameter configuration.

A Appendix

A.1 Datasets

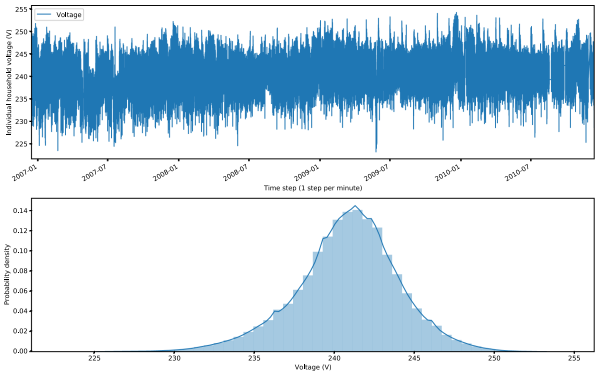

Fig. 4. Top - The individual household voltage dataset. Bottom - Probability distribution of the voltage dataset.

Table 7. Summary of the basic statistics of the segmented datasets and the input vector size per feature type.

	Voltage	Methane	NYSE	JSE
Number of local raw data points	2075259	41786	13563	3094
Number of trend lines	42280	4419	10015	1001
Mean ± deviation of the trend slope	-0.21 ± 10.41	0.17 ± 18.12	5.44 ± 81.27	0.21 ± 18.18
Mean ± deviation of the trend duration	50.08 ± 60.36	10.46 ± 67.03	2.35 ± 0.81	4.09 ± 5.23
Raw local data feature size	19	100	4	2
Raw local data + Trend line feature size	21	102	6	4
Number of data instances	42279	4418	10014	1001

A.2 Model Update with Warm-Start

The walk-forward evaluation procedure requires as many training episodes as the number of splits: one initial training and many model updates. This many training episodes can be computationally very expensive, particularly for deep neural networks. Thus, in this work, model update with warm start initialisation is used to reduce the training time of the neural network based algorithms.

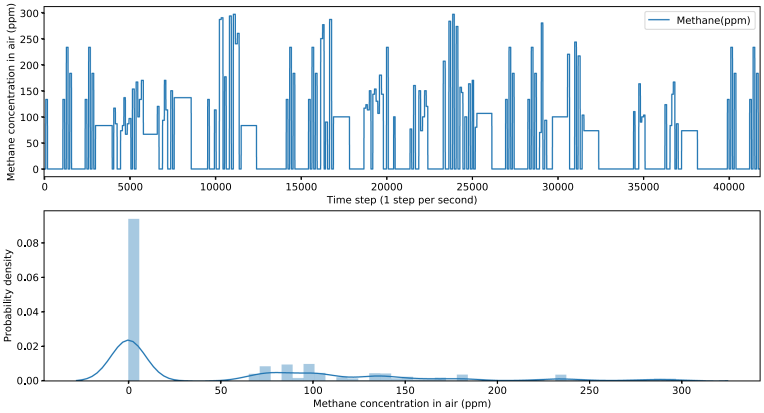

Fig. 5. Top - Methane concentration in air over time. Bottom - Probability distribution of the methane dataset.

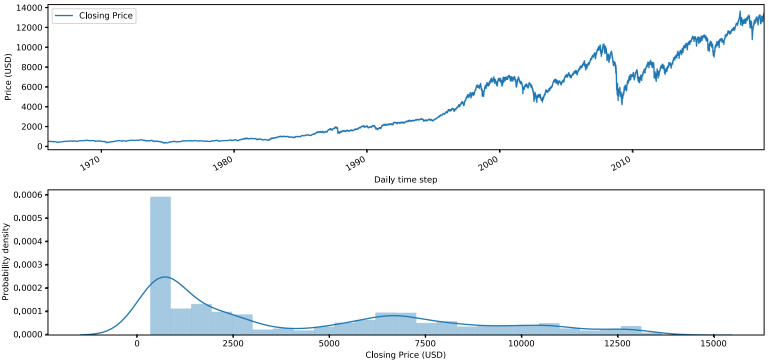

Fig. 6. Top - The composite New York Stock Exchange (NYSE) closing price dataset. Bottom - Probability distribution of the NYSE dataset.

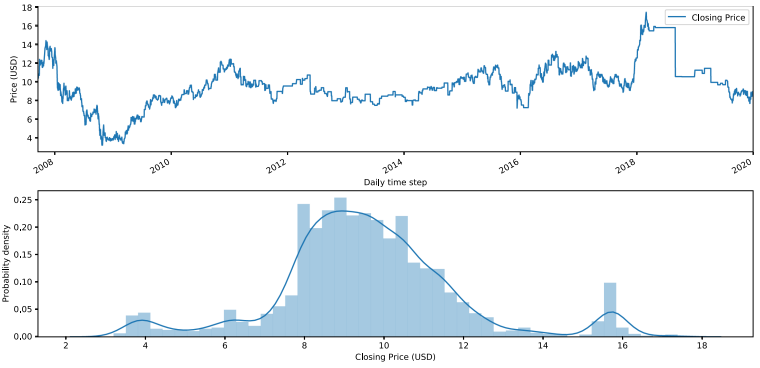

Fig. 7. Top - Composite Johannesburg Stock Exchange (JSE) closing price dataset. Bottom - Probability distribution of the JSE dataset.

Table 8. Summary of the data instance partitioning

	Voltage	Methane	NYSE	JSE
Number of data instances	42279	4418	10014	1001
Chosen total test sets percentage	80%	10%	50%	10%
Chosen test set size	4227	10	1001	1
Number of splits	8	44	5	101
Validation set size	4227	10	1001	1
Training set size	4227	3967	4008	899

That is, during model update, the new network is initialised with the weights of the previous model. In effect, the patterns learnt by the previous network are transferred to the new model, therefore, reducing the number of epochs required to learn the new best function. In practice, the walk-forward evaluation with warm start corresponds to performing the first training with the maximum number of epochs required to converge, then using a fraction of this number for every other update. This fraction - between 0.0 and 1.0 - becomes an additional hyperparameter dubbed *warm start*. The lowest value that out-performed the model update without warm-start is used as the best value, because this technique is essentially used to speed-up the model updates.

The speed-up, i.e. the expected reduction factor in the total number of epochs can be computed in advance using Eq. 4. The Eq. 4 is derived from Eq. 2 and Eq. 3.

$$E' = E + E \times (S - 1) \times \omega \tag{2}$$

$$E' = E \times (1 + (S - 1) \times \omega) \tag{3}$$

$$\textit{speed-up} = \frac{E}{E'} = \frac{S}{1 + (S - 1) \times \omega} \tag{4}$$

Where, $E' \rightarrow$ *Total epochs with warm start*, $E \rightarrow$ *Epochs per split without warm-start*, $S \rightarrow$ *Number of data partition splits*, $\omega \rightarrow$ *warm-start fraction.*

A.3 Best Hyperparameters

Table 9. Our best TreNet hyperparameters found by manual experimentation. "?" means *unknown* and $S = \{300, 600, 900, 1200\}$

	Dropout	L2	LR	LSTM cells	CNN filters	Fusion layer	Batch Size	Epochs	Warm start
Voltage	0.0	5e−4	1e−3	[600]	[16, 16]	300	2000	100	0.2
Methane	0.0	5e−4	1e−3	[1500]	[4, 4]	1200	2000	2000	0.1
NYSE	0.0	0.0	1e−3	[600]	[128, 128]	300	5000	100	0.5
JSE	0.0	0.0	1e−3	[5]	[32, 32]	10	500	100	0.05
Lin et al. [16]	0.5	5e−4	?	[600]	[32, 32]	$From\ S$	?	?	*N/A*

A.4 Additional Results

Table 10. Hyperparameters optimised for the vanilla DNN algorithms and their best values found for each dataset

		Voltage	Methane	NYSE	JSE
MLP	Batch size	4000	250	5000	250
	Warm start	0.1	0.1	0.7	0.05
	Learning rate	1e−4	1e−3	1e−3	1e−3
	Dropout	0.0	0.0	0.0	0.0
	Weight decay	0.0	0.0	5e−4	0.0
	Number of epochs	10000	15000	500	100
	Layer configuration	[500, 400, 300]	[500, 400]	[500, 400, 300]	[100]
LSTM	Batch size	4000	2000	5000	1000
	Warm start	0.1	0.1	0.01	0.05
	Learning rate	1e−2	1e−4	1e−3	1e−3
	Dropout	0.0	0.0	0.5	0.5
	Weight decay	0.0	0.0	5e−5	0.0
	Number of epochs	1000	15000	100	100
	Cell configuration	[600]	[600, 300]	[100]	[100]
CNN	Batch size	2000	250	5000	1000
	Warm start	0.5	0.3	0.4	0.1
	Learning rate	1e−3	1e−3	1e−3	1e−3
	Dropout	0.0	0.0	0.0	0.0
	Weight decay	5e−5	5e−4	0.0	0.0
	Number of epochs	15000	1000	12000	100
	Filter configuration	[16]	[32, 32]	[32]	[32, 32]
	Kernel configuration	[2]	[2, 4]	[1]	[1, 1]
	Pooling type	Max	Max	Identity	Identity
	Pooling size	2	5	N/A	N/A

Table 11. Best hyperparameters of the traditional ML algorithms.

Algorithm	Hyperparameter	Voltage	Methane	NYSE	JSE
RF	Number of estimators	50	50	200	100
	Maximum depth	2	10	1	1
	Bootstrap	2000	False	True	False
	Warm start	False	False	True	True
GBM	Bootstrap type	gbdt	gbdt	gbdt	gbdt
	Number of estimators	1	10000	1	4
	Learning rate	2000	0.1	0.2	0.1
SVR	Gamma	0.1	1e−4	1e−1	1e−4
	C	4	10000	100	500

Table 12. Performance of vanilla DNN algorithms on raw data alone and raw data and trend line features.

	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	9.04 ± 0.06	62.82 ± 0.04	35.93 ± 0.05	14.57 ± 0.10	49.79 ± 4.85	32.18 ± 2.47
Raw data + Trend lines	**9.03 ± 0.06**	62.81 ± 0.04	**35.92 ± 0.05**	**14.56 ± 0.19**	**34.46 ± 2.79**	**24.51 ± 1.49**
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	90.76 ± 4.43	33.08 ± 42.08	61.92 ± 23.26	**19.87 ± 0.01**	**12.51 ± 0.09**	**16.19 ± 0.05**
Raw data + Trend lines	90.45 ± 2.55	25.34 ± 24.09	57.90 ± 13.32	21.13 ± 0.30	12.59 ± 0.14	16.86 ± 0.22
MLP						
	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	10.30 ± 0.0	62.87 ± 0.0	36.59 ± 0.0	**14.21 ± 0.19**	56.37 ± 1.77	35.29 ± 0.68
Raw data + Trend lines	10.30 ± 0.0	62.87 ± 0.0	36.59 ± 0.0	14.77 ± 0.51	**48.03 ± 5.74**	**31.40 ± 3.13**
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	86.56 ± 0.01	**0.41 ± 0.08**	**43.49 ± 0.05**	**19.83 ± 0.01**	**12.68 ± 0.01**	**16.26 ± 0.01**
Raw data + Trend lines	**86.50 ± 0.01**	0.47 ± 0.03	**43.49 ± 0.02**	20.16 ± 0.03	12.74 ± 0.02	16.45 ± 0.03
LSTM						
	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	**9.24 ± 0.10**	**62.40 ± 0.13**	**35.82 ± 0.12**	15.07 ± 0.35	54.79 ± 4.55	34.93 ± 2.45
Raw data + Trend lines	33.26 ± 19.41	90.78 ± 53.17	62.02 ± 36.29	15.14 ± 0.28	**37.92 ± 4.11**	**26.53 ± 2.20**
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	89.31 ± 1.38	12.21 ± 12.17	50.76 ± 6.78	**19.90 ± 0.06**	**12.48 ± 0.21**	**16.19 ± 0.14**
Raw data + Trend lines	90.44 ± 1.74	14.05 ± 9.52	52.25 ± 5.63	21.41 ± 0.33	12.71 ± 0.15	17.06 ± 0.24
CNN						

Table 13. Performance of traditional ML algorithms on raw data alone and raw data and trend line features.

	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Local raw data	9.53 ± 0.0	**63.11 ± 0.20**	36.32 ± 0.10	**10.09 ± 0.01**	20.79 ± 0.01	**15.44 ± 0.01**
Local raw data + Trend lines	**9.35 ± 0.0**	63.19 ± 0.29	**36.27 ± 0.15**	11.53 ± 0.0	**20.73 ± 0.01**	16.13 ± 0.01
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Local raw data	88.75 ± 0.17	**0.29 ± 0.0**	44.52 ± 0.09	**20.21 ± 0.0**	**12.67 ± 0.0**	**16.44 ± 0.0**
Local raw data + Trend lines	86.53 ± 0.01	0.41 ± 0.0	**43.47 ± 0.01**	22.68 ± 0.0	12.69 ± 0.0	17.69 ± 0.0
RF						
	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Local raw data	**10.0 ± 0.0**	62.67 ± 0.0	36.34 ± 0.0	13.05 ± 0.0	75.10 ± 0.0	44.08 ± 0.0
Local raw data + Trend lines	10.01 ± 0.0	**62.63 ± 0.0**	**36.32 ± 0.0**	**12.02 ± 0.0**	**38.34 ± 0.0**	**25.18 ± 0.0**
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Local raw data	86.62 ± 0.0	0.42 ± 0.0	43.52 ± 0.0	20.08 ± 0.0	**12.62 ± 0.0**	16.35 ± 0.0
Local raw data + Trend lines	**86.42 ± 0.0**	**0.41 ± 0.0**	**43.42 ± 0.0**	**19.93 ± 0.0**	12.65 ± 0.0	**16.29 ± 0.0**
GBM						
	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	**9.32 ± 0.0**	**62.58 ± 0.0**	**35.95 ± 0.0**	14.98 ± 0.0	**34.39 ± 0.0**	**24.69 ± 0.0**
Raw data + Trend lines	9.54 ± 0.0	62.62 ± 0.0	36.08 ± 0.0	17.95 ± 0.0	34.52 ± 0.0	26.24 ± 0.0
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	86.55 ± 0.0	0.42 ± 0.0	**43.49 ± 0.0**	**20.01 ± 0.0**	**12.85 ± 0.0**	**16.43 ± 0.0**
Raw data + Trend lines	**86.54 ± 0.0**	0.45 ± 0.0	43.50 ± 0.0	23.27 ± 0.0	13.19 ± 0.0	18.23 ± 0.0
SVR						

References

1. Bergmeir, C., Benítez, J.M.: On the use of cross-validation for time series predictor evaluation. Inf. Sci. **191**, 192–213 (2012). https://doi.org/10.1016/j.ins.2011.12.028
2. Chang, L., Chen, P., Chang, F.: Reinforced two-step-ahead weight adjustment technique for online training of recurrent neural networks. IEEE Trans. Neural Netw. Learn. Syst. **23**(8), 1269–1278 (2012)
3. Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR abs/1412.3555 (2014)
4. Falkner, S., Klein, A., Hutter, F.: BOHB: robust and efficient hyperparameter optimization at scale. In: Proceedings of Machine Learning Research, PMLR, Stockholmsmässan, Stockholm Sweden, vol. 80, pp. 1437–1446, 10–15 July 2018. http://proceedings.mlr.press/v80/falkner18a.html
5. Feng, F., Chen, H., He, X., Ding, J., Sun, M., Chua, T.S.: Enhancing stock movement prediction with adversarial training. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 7 July 2019, pp. 5843–5849. https://doi.org/10.24963/ijcai.2019/810

6. Guo, J., Li, X.: Prediction of index trend based on LSTM model for extracting image similarity feature. In: Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, pp. 335–340. AICS 2019. ACM, New York (2019). https://doi.org/10.1145/3349341.3349427
7. Guo, T., Xu, Z., Yao, X., Chen, H., Aberer, K., Funaya, K.: Robust online time series prediction with recurrent neural networks. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 816–825 (2016). https://doi.org/10.1109/DSAA.2016.92
8. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on ImageNet classification (2015)
9. Kara, Y., Acar Boyacioglu, M., Baykan, Ö.K.: Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul Stock Exchange. Expert Syst. Appl. **38**(5), 5311–5319 (2011). https://doi.org/10.1016/j.eswa.2010.10.027
10. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proceedings 2001 IEEE International Conference on Data Mining, pp. 289–296 (2001). https://doi.org/10.1109/ICDM.2001.989531
11. Keogh, E., Pazzani, M.: An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: KDD, vol. 98, pp. 239–243 (1998). https://doi.org/10.1.1.42.1358. http://www.aaai.org/Papers/KDD/1998/KDD98-041.pdf
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014)
13. Kumar, I., Dogra, K., Utreja, C., Yadav, P.: A comparative study of supervised machine learning algorithms for stock market trend prediction. In: 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), pp. 1003–1007 (2018)
14. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
15. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: a novel bandit-based approach to hyperparameter optimization. J. Mach. Learn. Res. **18**(1), 6765–6816 (2017)
16. Lin, T., Guo, T., Aberer, K.: Hybrid neural networks for learning the trend in time series. In: IJCAI - Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 2273–2279 (2017). https://doi.org/10.24963/ijcai.2017/316. https://www.ijcai.org/proceedings/2017/316
17. Liu, Q., Cheng, X., Su, S., Zhu, S.: Hierarchical complementary attention network for predicting stock price movements with news. In: Proceedings of the 7th ACM International Conference on Information and Knowledge Management, CIKM 2018, pp. 1603–1606. ACM, New York (2018). https://doi.org/10.1145/3269206.3269286. http://doi.acm.org/10.1145/3269206.3269286
18. Luo, L., Chen, X.: Integrating piecewise linear representation and weighted support vector machine for stock trading signal prediction. Appl. Soft Comput. **13**(2), 806–816 (2013). https://doi.org/10.1016/j.asoc.2012.10.026. http://www.sciencedirect.com/science/article/pii/S1568494612004796
19. Matsubara, Y., Sakurai, Y., Faloutsos, C.: AutoPlait: automatic mining of co-evolving time sequences. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD 2014, pp. 193–204. Association for Computing Machinery, New York (2014). https://doi.org/10.1145/2588555.2588556

20. Nelson, D.M., Pereira, A.C., De Oliveira, R.A.: Stock market's price movement prediction with LSTM neural networks. In: Proceedings of the International Joint Conference on Neural Networks (DCC), May 2017 pp. 1419–1426 (2017). https://doi.org/10.1109/IJCNN.2017.7966019
21. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
22. Sharma, N., Juneja, A.: Combining of random forest estimates using LSboost for stock market index prediction. In: 2017 2nd International Conference for Convergence in Technology (I2CT), pp. 1199–1202 (2017)
23. Wang, P., Wang, H., Wang, W.: Finding semantics in time series. In: Proceedings of the 2011 International Conference on Management of Data - SIGMOD 2011, p. 385 (2011). https://doi.org/10.1145/1989323.1989364. http://portal.acm.org/citation.cfm?doid=1989323.1989364
24. Wen, M., Li, P., Zhang, L., Chen, Y.: Stock market trend prediction using high-order information of time series. IEEE Access **7**, 28299–28308 (2019). https://doi.org/10.1109/ACCESS.2019.2901842. https://ieeexplore.ieee.org/document/8653278/
25. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 947–956. Association for Computing Machinery, New York (2009). https://doi.org/10.1145/1557019.1557122
26. Zhang, J., Cui, S., Xu, Y., Li, Q., Li, T.: A novel data-driven stock price trend prediction system. Expert Syst. Appl. **97**, 60–69 (2018). https://doi.org/10.1016/j.eswa.2017.12.026
27. Zhao, Y., Shen, Y., Zhu, Y., Yao, J.: Forecasting wavelet transformed time series with attentive neural networks. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 1452–1457 (2018)