An analysis of deep neural networks for predicting trends in time series data*

Kouame Kouassi[1] and Deshendran Moodley[2]

[1] University of Cape Town, Cape Town, South Africa
ksskou001@myuct.ac.za
[2] University of Cape Town, Cape Town, South Africa
deshen@cs.uct.ac.za

Abstract. The emergence of small and portable smart sensors have opened up new opportunities for many applications, including automated factories, smart cities and connected healthcare, broadly referred to as the Internet of Things (IoT). These devices produce time series data. While deep neural networks (DNNs) has been widely applied to computer vision, natural language processing and speech recognition, there is limited research on DNNs for time series prediction. Machine learning (ML) applications for time series prediction has traditionally involved predicting the next value in the series. However, in certain applications, segmenting the time series into a sequence of trends and predicting the next trend is preferred. Recently, a hybrid DNN algorithm, TreNet was proposed for trend prediction. TreNet, which combines an LSTM that takes in trendlines and a CNN that takes in point data was shown to have superior performance for trend prediction when compared to other approaches. However, the study used a standard cross-validation method which does not take into account the sequential nature of time series. In this work, we reproduce TreNet using a walk-forward validation method, which is more appropriate to time series data. We compare the performance of the hybrid TreNet algorithm, on the same three data sets used in the original study, to vanilla MLP, LSTM, and CNN that take in point data, and also to traditional ML algorithms, i.e. the Random Forest (RF), Support Vector Regression and Gradient Boosting Machine. Our results differ significantly from those reported for the original TreNet. In general TreNet still performs better than the vanilla DNN models, but not substantially so as reported for the original TreNet. Furthermore, our results show that the RF algorithm performed substantially better than TreNet on the methane data set.

Keywords: Time Series · Trend Prediction · Ensemble Methods · Automated Machine Learning. Bayesian Optimisation and HyperBand

1 Introduction

Theoretically, trend prediction is achievable by performing a multistep-ahead prediction and then fitting the predicted values to estimate the trend [4,17].

* Supported by the Centre for Artificial Intelligence Research.

This approach suffers from accumulative prediction errors [1,2], and is non-trivial [24]. An alternative approach is the use of pattern-based hidden Markov models (p-HMM)s to discover a pre-defined number of states and model the transition dynamics between these states [20,25] using the training set. The next trend is then predicted by first identifying the current state, which is used to perform the inference. HMMs only maintain short-term state dependences because of the memoryless Markov property. Furthermore, specifying the number of states requires task specific knowledge [17]. The multistep-ahead prediction and the p-HHM predict the trend in an implicit fashion.
In 2017, Lin et al. [17] 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 networks dubbed TreNet. The hybrid nature of TreNet is not only due to its use of both LSTM, and CNN; but also because, it leverages both *raw data* and *trend line* features. TreNet outperformed SVR, CNN, LSTM, pHHM [25], and cascaded CNN and RNN [17]. However, it presents some limitations. The first limitation is that time is not dealt with adequately in TreNet. 10-fold cross-validation with random shuffling is used for model development, and a single held-out set is used for testing [17]. The use of cross-validation with shuffling implies that data instances, which are generated after the given validation set, are used for training. This is technically incorrect because during inference only the past data instances are available. Besides, the use of a single held-out set is not suitable for non-stationary [3]. Furthermore, many important implementation details are not stated explicitly, which affects the reproducibility of the approach. For instance, the segmentation method used to transform the raw time series into trend lines is not apparent. Finally, ensemble regression models such as random forests (RF), and Gradient Boosting Machines (GBM) - which are very widely used traditional machine learning models [14,23] - are not included in the baseline algorithms. This paper addresses the highlighted shortcomings. Our research questions are:

1. Does the hybrid deep neural networks approach for trend prediction perform better than vanilla deep neural networks?
2. Do deep learning (DL) models perform better for trend prediction than simpler traditional machine learning (ML) models?

In the remaining of this paper, we first present the related work, and give a brief background on time series trend prediction. Then, we describe the experimental design, and present the results, which are discussed before concluding.

2 Related work

Traditional trend prediction approaches include Hidden Markov Models (HMM)s [25,20] and multi-step ahead predictions [4]. Recently, neural networks especially RNN-LSTMs and CNN [15] have become more prominent [16,8,17,5]. Combining the strenghts of both CNN and LSTM, Lin et al. [17] proposed TreNet to learn the long-term dependencies from trend lines and local selient features from local

raw data. Other works have added attention mechanisms to these architectures [28,6]. Zhao et al. [28] used wavelet transformed time series as features. Although these more recent work build on Lin et al. [17] initial work, they do not deal with trend prediction specifically. The stock market direction movement is a very active and related area of research [6,26,7,18,21,10]. However, this approach only tackles the direction of the time series, it does not predict the strength and the duration of the time series. Generally, the baseline methods used by prior works include neural networks, the naive last value prediction, ARIMA, SVR [17,27]. They do not include ensemble methods such as random forest, which are widely used particularly for stock market movement prediction[14,23].

3 Time series trend prediction

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.

3.1 Problem formulation

We define a continuous valued time series $X = \{x_1, ..., x_T\}$, the non-overlapping, time-ordered, successive K *trends* $\mathcal{T} = \{< s_k, l_k >\}$, and their corresponding *local data* $\mathcal{L} = \{< x_{t_k-w}, ..., x_{t_k} >\}$ of window w. A *trend* $< s_k, l_k >$ denotes a linear approximation by regression or by interpolation over a segment of X. l_k represents the *duration* and is given by the number of data points covered by the trend k; thus, $\sum_{k=1}^{K} l_k = T + K - 1$. The slope s_k represents the *gradient* of the linear approximation given as an angle between -90 and 90. The *local data* $< x_{t_k-w}, ..., x_{t_k} >$ of the trend k is the local raw data that affects the evolving trend $< s_k, l_k >$ where t_k is the starting time of the trend k in $\mathcal{T}$.
The aim is to predict the *next trend* $< s_{k+1}, l_{k+1} >$ for a univariate time series.

3.2 Trend prediction process

The time series trend prediction process is shown in figure 1. It starts with acquiring a set of the time series in question. The dataset is then segmented into piecewise linear approximations. The output of the segmentation process is a sequence of trend lines and their corresponding raw local data. The data instances, i.e. the input-output pairs are obtained from this sequence using the sliding window technique. The data instances are partitioned in an overlapping training-validation-test fashion. The algorithm selection and hyperparameter optimisation process finds the optimal algorithm and its optimal hyperparameter

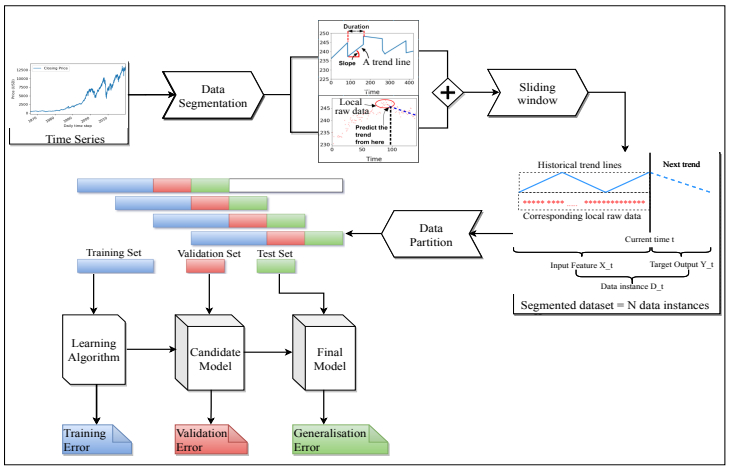

Fig. 1: Overview diagram of trend prediction process

configuration. More specifically, this process selects a supervised learning algorithm, which is trained using the training sets. The output of the training process is a candidate model, which is evaluated using the validation sets. This evaluation is used to select the optimal model. The generalisation ability of the final model, i.e. the optimal model, is estimated on the test sets.

3.3 Datasets

We performed our experiments with the four datasets described below.

1. *The voltage dataset* from the UCI machine learning repository[3]. 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. These properties can be observed in figure 4 - in the appendices. It corresponds to the power consumption dataset used by Lin et al. [17].
2. *The methane dataset* from the UCI machine learning repository[4]. We used a resampled set of size of 41786 at a frequency of 1Hz. The methane dataset is skewed to the right of its mean value and exhibits very sharp changes with medium to low volatility. These properties can be observed in figure 5 - in the appendices. It is corresponds to the gas sensor dataset used by Lin et al. [17].
3. *NYSE dataset* from Yahoo finance[5]. 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. These properties can be observed in figure 7 - in the appendices. It corresponds to the stock market dataset used by Lin et al. [17].

4. *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. These properties can be observed in figure 8 - in the appendices.

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

Table 1: Summary of the characteristics of the datasets.

	Seasonality	***Periodicity***	***Skewness***	***Volatility***
Voltage	seasonal	yearly	symmetric	very high
Methane	non-seasonal	N/A	right skewness	medium to low
NYSE	non-seasonal	N/A	right skewness	low to high
JSE	non-seasonal	N/A	almost symmetric	medium to low

3.4 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 [11,12], similar to Wang et al. [25]. The data instances, i.e. the input-output pairs are formed using the sliding window technique. The output is the next trend $T_{k+1} =< s_{k+1}, l_{k+1} >$, i.e. the subsequent trend to the trend $T_k =< s_k, l_k >$, which covers the current time t.
The input features of the hybrid neural networks consists of the current trend $T_k =< s_k, l_k >$ - fed into the LSTM - and the w local raw data points $L_k =< x_{t_k-w}, ..., x_{t_k} >$ - fed into the CNN. For the ensemble methods, we experiment with three different types: only the *local raw data points* L_k, the current *trend* T_k, and the concatenation of the current *trend and the local raw data points* $T_k + L_k =< s_k, l_k, x_{t_k-w}, ..., x_{t_k} >$.
The window size w is determined by the duration of the first trend line. The appendix table 6 provides a descriptive summary of the segmented datasets and the input-output data instances.

3.5 Data partitioning

The input-output data instances are partitioned into training, validation, and test sets in a successive and overlapping fashion [19]. Similar to Lin et al. [17],

[3] https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption
[4] https://archive.ics.uci.edu/ml/datasets/gas+sensor+array+under+dynamic+gas+mixtures
[5] https://finance.yahoo.com

the test sets proportion is set to about 10% for the methane and JSE datasets. For the voltage and NYSE dataset, it is set to about 80% and 50%, respectively. This exception is because their relatively greater numbers of data instances allow for higher percentages, and still have a resonable training set size. The validation set size is made equal to the test set. The partition sizes for each dataset are given in the appendix table 7.

The *number of splits*, the *maximum training set size* are respectively determined using the the equation 1 and the equation 2. $\alpha \in (0, 1)$ and represents the percentage of the total data instances used for hold-out evaluation, that is $T_{total\,test}$.

$$Number\ of\ splits = S = \lfloor \frac{T_{total\,test}}{T_{test}} \rfloor = \lfloor \frac{\alpha \times T_{total}}{T_{test}} \rfloor \tag{1}$$

$$T_{training_{max}} = \lfloor T_{total} - T_{validation} - S \times T_{test} \rfloor = \lfloor T_{total} - (S+1) \times T_{test} \rfloor \tag{2}$$

These equations are derived and further explained in the appendix 7.2. For the voltage dataset, the used training set size is made equal to the test set, resulting in the last six data instances being unused. For the methane dataset, it is set to 3967 resulting in the last data instance being unused.

3.6 Model evaluation

The walk-forward evaluation procedure, with the successive and overlapping training-validation-test partition, is used to evaluate the performance of the models. The root mean square error (RMSE) is used as evaluation metric. The equally weighted average of the slope RMSE and the duration RMSE is used as a single metric, where applicable. For each partition, the different splits are concatenated into a single series and used to calculate the overall RMSE using equation 3.

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - y_t')^2} \tag{3}$$

The walk-forward evaluation procedure requires as many training episodes as the number of splits: the initial training and the subsequent model updates. For DL models, the neural networks are initialised using the weights of most recent model, during model update. This makes the training of the network faster without compromising its generalisation ability. More details about this technique is given in the appendix section 7.3 and section 7.4.

Each experiment is run 10 times because of the stochastic nature some algorithms, which makes them sensible to random initialisation. The results therefore consists of the mean and the standard deviation of these 10 runs, where applicable.

3.7 Learning algorithm design

This study considered seven different algorithms to predict time series trends. These algorithms consist of the hybrid neural networks dubbed TreNet [17], the multilayer perceptrons (MLP), the long short-term memory recurrent neural networks (LSTM-RNN), the convolutional neural networks (CNN), the random forest (RF) regressor, the gradient boosting machine (GBM) regressor, and the radial-based support vector regressor (SVR). TreNet, MLP, LSTM, and CNN are neural networks based algorithms; RF and GBM are ensemble methods; and SVR is a kernel based algorithm. The ensemble and kernel methods are traditional ML algorithms.

3.8 Deep learning (DL) model design

The TreNet model is similar to the initial architecture [17] proposed by Lin et al. [17] as shown in figure 2. The LSTM consisted of a single LSTM layer. The

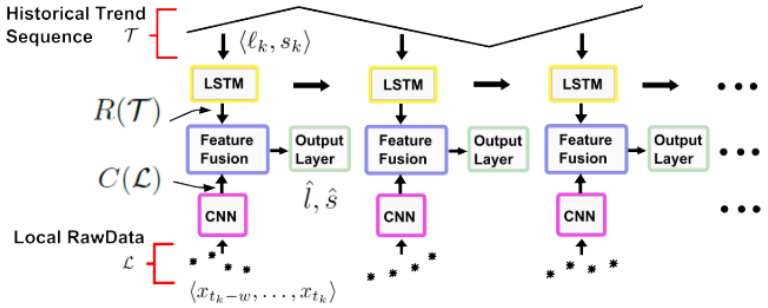

Fig. 2: Illustration of the hybrid neural network architecture [17]

CNN is composed of two stacked [17] 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 consisted 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.

We manually tuned the *dropout rate*, *weight decay*, the *learning rate*, the *number of LSTM cells*, the *CNN filter sizes*, the *number of fusion layer neurons*, the *batch size*, the *number of epochs*, and the *warm start* hyperparameters on the validation sets. The optimal hyperparameters for each dataset are shown in the appendix table 8.

The MLP model model consists of N number of fully connected neural network layers, where, $N \in [1, 5]$. Each layer is followed by a ReLU activation function to capture non-linear patterns in the data. 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 weights of the network are initialised using the He initialisation technique [9] with normal distribution.

The LSTM model consists of N LSTM layers, where $N \in [1, 3]$. Each LSTM is followed by a ReLU activation function to extract non-linear patterns, and a dropout layer to prevent overfitting. After the last LSTM layer, a fully connected neural network layer is added. This layer takes the feature representation extracted by the LSTM layers as its input and predicts the next trend. Similar to the MLP, this fully connected layer is initialised with the He initialisation [9]. To learn long-term relationships between trends, the LSTM layers are not re-initialised at every epoch.

The CNN model 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 one layer CNN is illustrated in figure 3. This architecture is inspired by Lin et al's implementation [17]. Both the convolutional

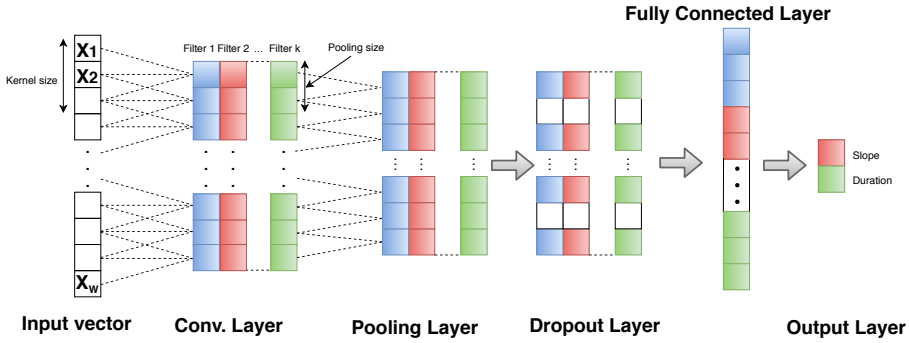

Fig. 3: The structure of a one layer 1D-convolution neural network.

and fully connected layers are initialised with the He initialisation technique [9].

DL model training, and regularisation The equally weighted average slope and duration mean square error (MSE) is used as loss function when training the

neural network based algorithms. Adam optimizer [13] is used for learning the optimal weights of the neural networks. It is selected because it achieves good results faster than other methods [13]. To prevent overfitting, dropout and L2 regularisation (weight decay) are used [17]. The actual value per algorithm per dataset is determined through tuning.
To make the neural networks robust to random initialisation, their weights are initialised using the He initialisation technique [9] with normal distribution. The ReLU function is selected as the non-linear function. The mode is set to *fan-in*, that is the magnitude of the variance of the weights in the forward pass is preserved.

3.9 Traditional machine learning (ML) model design

SVR and RF are implemented using Sklearn [22], but, GBM is implemented with LightGBM[6]. 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 optimal hyperparameter configurations per dataset are shown in the appendix table 9. We use the MSE loss for the RF.

4 Overview of experiments

We performed three experiments on the four datasets described in section 3.3. In experiment 1, we implement and evaluate a recent hybrid deep learning (DL) trend prediction approach, i.e. TreNet [17]. TreNet uses a hybrid deep learning structure, that combines both an LSTM and a CNN, 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 DL and non-DL approaches.

4.1 Replicating TreNet

The TreNet approach, recently proposed by Lin et al., combines an LSTM and a CNN into a hybrid neural network [17]. 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

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

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 [17]. A walk-forward validation with overlapping partition [19] is better suited for evaluating and comparing model performance on time series data [19]. Since this brings into question the veracity of the reported results, we attempted to replicate the TreNet approach using a walk forward validation over random shuffling and cross validation.

In order to compare our results with the original TreNet we use a similar performance measure to Lin et al. [17]. 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" [17]. The use of a relative metric makes comparison easier, since the RMSE is scale-dependent, and the trendlines generated in this study may differ from Lin et al.'s [17]. Lin et al. [17] 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.

Table 2: Comparison of the RMSE achieved by our hybrid neural network's performance and Lin et al.'s results.

	Voltage			Methane			NYSE		
	Slope	Duration	Average	Slope	Duration	Average	Slope	Duration	Average
Our naive model	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 % improvement	**45.87**	27.90	30.87	**47.90**	**79.56**	**74.58**	**31.67**	−272.73	**30.89**
Lin et al.'s naive model	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 % improvement	39.11	**35.43**	**36.71**	10.50	4.69	5.63	23.31	**25.09**	24.27

As shown in Table 2 the results of our experiment differ substantially from those reported for the original TreNet. Our TreNet models' percentage improvement over the naive last value prediction is 13.25 (**74.58**/5.63) and 1.27 (**30.89**/24.27) times greater than Lin et al.'s [17], on the methane and NYSE datasets respectively; but 1.19 (36.71/**27.90**) times smaller on the voltage dataset. The naive last value prediction model 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 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 on 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 a validation/test set, cross-validation - used by Lin et al. [17] - takes all the remaining dataset for training, which contains data instances generated by a different process. Thus, the network learns long-range relationships that are not useful for the current test set.

Although the use of cross-validation may not be detrimental to non-evolving time series such as the voltage dataset, it mis-represents the generalisation ability of the model during model development. This is because a given validation set (except the last one, in order of generation), cross-validation uses data instances that are generated after the given validation set for training. This is not technically correct because, during inference only the past data instances are available. Regardless of this model development weakness in their approach, our model had a worse improvement on the voltage dataset. This is attributed to our use of a smaller window size for the *local raw data* fed into the CNN. We used 19 compared to their optimal value of 700 on the voltage dataset.

This shows 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 optimal value. The other weakness concerns the use of a sampled version of the methane dataset instead of the complete methane dataset.

4.2 Vanilla DL models

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

Table 3 shows the average RMSE values for slope and trend predictions achieved by the MLP, LSTM, CNN and the 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 DL model over TreNet. The best model is chosen based on the overall comparison metric.

In general TreNet still performs better than the vanilla DL models, but does not outperform the vanilla models on all the datasets. The most noticeable case is on

Table 3: Comparison of TreNet with the vanilla DL models

	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**
% Improvement	-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**
% Improvement	**0.38**	**66.67**	**1.29**	-1.12	-0.16	-0.75

the NYSE, where the LSTM model outperforms the TreNet model on both the slope and duration prediction. This contradicts Lin et al. [17]'s findings, where TreNet clearly outperforms all other models including LSTM. On average, Lin et al.'s [17] TreNet model outperformed their LSTM model by 22.48%; whereas, our TreNet implementation underperformed our LSTM model by 1.31%. However, Lin et al. [17]'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. [17]. 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.

4.3 Traditional ML models

Given the new validation method, we now compare the performance of DL 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. [17] 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 DL models on each dataset. The best DL 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 DL model, where, the best model is selected based on the equally weighted average slope and duration RMSE, i.e. average.

Table 4: Comparison of the best DL models (Best DL) with the traditional ML models (Trad. ML).

	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
RF	9.53 ± 0.0	63.11 ± 0.20	36.32 ± 0.10	$\mathbf{10.09 \pm 0.01}$	$\mathbf{20.79 \pm 0.01}$	$\mathbf{15.44 \pm 0.01}$
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 DL	$\mathbf{9.25 \pm 0.0}$	$\mathbf{62.37 \pm 0.01}$	$\mathbf{35.81 \pm 0.01}$	14.87 ± 0.40	31.25 ± 2.62	23.06 ± 1.51
% improvement	-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	$\mathbf{0.29 \pm 0.0}$	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	$\mathbf{86.55 \pm 0.0}$	0.42 ± 0.0	43.49 ± 0.0	20.01 ± 0.0	12.85 ± 0.0	16.43 ± 0.0
Best DL	86.56 ± 0.01	$\mathbf{0.41 \pm 0.08}$	43.49 ± 0.05	$\mathbf{19.65 \pm 0.05}$	$\mathbf{12.49 \pm 0.04}$	$\mathbf{16.07 \pm 0.05}$
% improvement	**0.01**	2.44	0.0	-2.19	-1.04	-1.74

The best traditional ML algorithms underperformed 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 DL model, i.e. TreNet by 33.04% on the methane dataset; while the SVR model matched the performance of the best DL 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 this dataset.
The fact that the radial-based SVR performed better than TreNet on the NYSE dataset contradicts Lin et al. [17]'s results. We attribute this to the use the use of *local raw data* features alone, instead of *local raw data* plus *trend line* features used by Lin et al. [17].

4.4 Summary - Best models from manual model selection

Table 5 shows the best models found from this manual process as well as their performance, i.e. the slope, duration, and average RMSE. The vanilla DL models do not appear to be the best models for trend prediction. However, they are good candidates for trend prediction with local raw data especially on non-stationary datasets. The next section of this paper performs automatic vanilla DL trend prediction model selection.

Table 5: Summary of the performance of the best trend prediction models

	Voltage	*Methane*	*NYSE*	*JSE*
Model	TreNet	RF	SVR/LSTM	TreNet
Slope	9.25 ± 0.0	10.09 ± 0.01	$86.55 \pm 0.0/86.56 \pm 0.01$	19.65 ± 0.05
Duration	62.37 ± 0.01	20.79 ± 0.01	$0.42 \pm 0.0/0.41 \pm 0.08$	12.49 ± 0.04
Average	35.81 ± 0.01	15.44 ± 0.01	$43.49 \pm 0.0/43.49 \pm 0.05$	16.07 ± 0.05

5 Discussions

In this section, the results and findings of the experiments are discussed. Directly predicting time series trends using trend lines as proposed by Lin et al. [17] is successfully replicated with best effort. This confirms that this approach is a promising avenue for trend prediction. The overall superiority of our implementation over Lin et al.'s [17] is attributed to the use of a more appropriate evaluation metric. The limitations of this replication are two-folds. Firstly, for each dataset the duration of the local raw data feature is determined by the lenght the first trend line. It is not tuned to select the optimal value. Secondly, a sampled version of the methane dataset is used instead of the complete methane dataset. These limitations do not affect our main findings - discussed below.

First, feature engineering more specifically trend line features can improve the performance of trend prediction models in most cases. This improvement is especially true for boosting algorithms such as the GBM, for which, adding the trend line features to the local raw data features improved the performance of the model.

Second, Neural networks based algorithms such as TreNet perform better than traditional ML algorithms on time series datasets that are (almost) normally distributed. However, traditional ML algorithms especially ensemble methods such as RF and GBM perform better on non-normally distributed time series datasets - such as datasets that are skewed to the right. In this case, it is better to use such simpler models, which are less complex and are less resource hungry.

6 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 walk-forward validation instead of the standard cross-validation. We compared TreNet to vanilla deep neural networks (DNNs) that take in point data features. Our results show that TreNet does not significantly outperform vanilla DNN models or can be beaten by an LSTM with point data features alone. Furthermore, we compared DNNs approach demonstrate empirically that for non-normally distributed datasets, traditional ML algorithms such as RF and GBM are simpler

and favourably competitive alternatives to neural networks. Finding the optimal model for a particular time series requires extensive experimentation by a machine learning expert, and often requires information about the characteristics of that time series. This work could be extended by automatically finding the optimal model for a given trend prediction problem..

References

1. Bao, Y., Xiong, T., Hu, Z.: Multi-step-ahead time series prediction using multiple-output support vector regression. Neurocomputing **129**, 482 – 493 (2014). https://doi.org/https://doi.org/10.1016/j.neucom.2013.09.010, http://www.sciencedirect.com/science/article/pii/S092523121300917X
2. Ben Taieb, S., Atiya, A.: A bias and variance analysis for multistep-ahead time series forecasting. IEEE Transactions on Neural Networks and Learning Systems **27**(1), 62–76 (1 2016). https://doi.org/10.1109/TNNLS.2015.2411629
3. Bergmeir, C., Benítez, J.M.: On the use of cross-validation for time series predictor evaluation. Inf. Sci. **191**, 192213 (May 2012). https://doi.org/10.1016/j.ins.2011.12.028, https://doi.org/10.1016/j.ins.2011.12.028
4. Chang, L., Chen, P., Chang, F.: Reinforced two-step-ahead weight adjustment technique for online training of recurrent neural networks. IEEE Transactions on Neural Networks and Learning Systems **23**(8), 1269–1278 (2012)
5. Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR **abs/1412.3555** (2014)
6. 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. pp. 5843–5849. International Joint Conferences on Artificial Intelligence Organization (7 2019). https://doi.org/10.24963/ijcai.2019/810, https://doi.org/10.24963/ijcai.2019/810
7. 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, NY, USA (2019). https://doi.org/10.1145/3349341.3349427, http://doi.acm.org/10.1145/3349341.3349427
8. Guo, T., Xu, Z., Yao, X., Chen, H., Aberer, K., Funaya, K.: Robust Online Time Series Prediction with Recurrent Neural Networks. 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA) pp. 816–825 (2016). https://doi.org/10.1109/DSAA.2016.92, http://ieeexplore.ieee.org/document/7796970/
9. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015)
10. 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 Systems with Applications **38**(5), 5311–5319 (2011). https://doi.org/10.1016/j.eswa.2010.10.027, http://dx.doi.org/10.1016/j.eswa.2010.10.027

11. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. Proceedings 2001 IEEE International Conference on Data Mining pp. 289–296 (2001). https://doi.org/10.1109/ICDM.2001.989531, http://ieeexplore.ieee.org/document/989531/
12. Keogh, E., Pazzani, M.: An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. Kdd **98**, 239–243 (1998). https://doi.org/10.1.1.42.1358, http://www.aaai.org/Papers/KDD/1998/KDD98-041.pdf
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
14. 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)
15. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
16. Li, P., Abdel-Aty, M., Yuan, J.: Real-time crash risk prediction on arterials based on lstm-cnn. Accident Analysis & Prevention **135**, 105371 (2020). https://doi.org/https://doi.org/10.1016/j.aap.2019.105371, http://www.sciencedirect.com/science/article/pii/S0001457519311108
17. Lin, T., Guo, T., Aberer, K.: Hybrid Neural Networks for Learning the Trend in Time Series. 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
18. Liu, Q., Cheng, X., Su, S., Zhu, S.: Hierarchical complementary attention network for predicting stock price movements with news. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 1603–1606. CIKM '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3269206.3269286, http://doi.acm.org/10.1145/3269206.3269286
19. Luo, L., Chen, X.: Integrating piecewise linear representation and weighted support vector machine for stock trading signal prediction. Applied Soft Computing **13**(2), 806 – 816 (2013). https://doi.org/https://doi.org/10.1016/j.asoc.2012.10.026, http://www.sciencedirect.com/science/article/pii/S1568494612004796
20. 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. p. 193204. SIGMOD '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2588555.2588556, https://doi.org/10.1145/2588555.2588556
21. Nelson, D.M., Pereira, A.C., De Oliveira, R.A.: Stock market's price movement prediction with LSTM neural networks. Proceedings of the International Joint Conference on Neural Networks **2017-May**(Dcc), 1419–1426 (2017). https://doi.org/10.1109/IJCNN.2017.7966019
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
23. 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)

24. Venkatraman, A., Hebert, M., Bagnell, J.A.: Improving multi-step prediction of learned time series models. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. p. 30243030. AAAI15, AAAI Press (2015)
25. Wang, P., Wang, H., Wang, W.: Finding semantics in time series. Proceedings of the 2011 international conference on Management of data - SIGMOD '11 p. 385 (2011). https://doi.org/10.1145/1989323.1989364, http://portal.acm.org/citation.cfm?doid=1989323.1989364
26. 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/
27. Zhang, J., Cui, S., Xu, Y., Li, Q., Li, T.: A novel data-driven stock price trend prediction system. Expert Systems with Applications **97**, 60–69 (2018). https://doi.org/10.1016/j.eswa.2017.12.026, https://doi.org/10.1016/j.eswa.2017.12.026
28. 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)

7 Appendices

7.1 Datasets

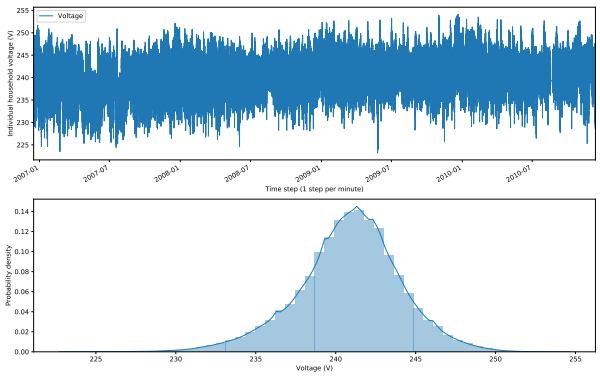

Fig. 4: Top - The individual household voltage dataset. Bottom - Probalility distribution of the voltage dataset.

7.2 Data partitioning

The data instances are split into training, validation, and test sets in a successive and overlapping fashion [19], as illustrated in figure 9. With this partitioning

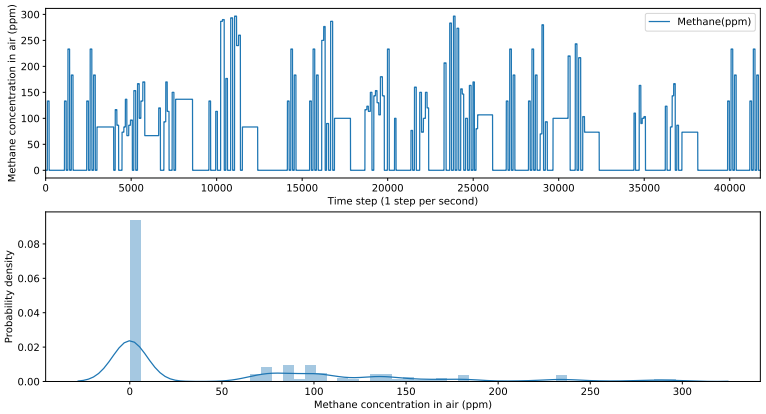

Fig. 5: Top - Methane concentration in air over time. Bottom - Probability distribution of the methane dataset.

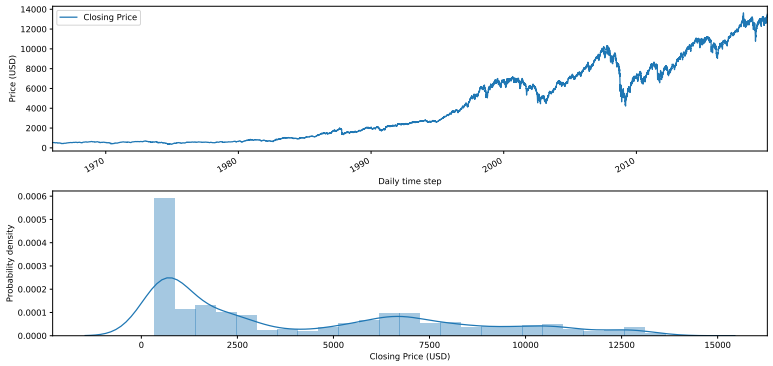

Fig. 6: Top - The composite New York Stock Exchange (NYSE) closing price dataset. Bottom - Probalility distribution of the NYSE dataset.

Fig. 7: Raw NYSE dataset.

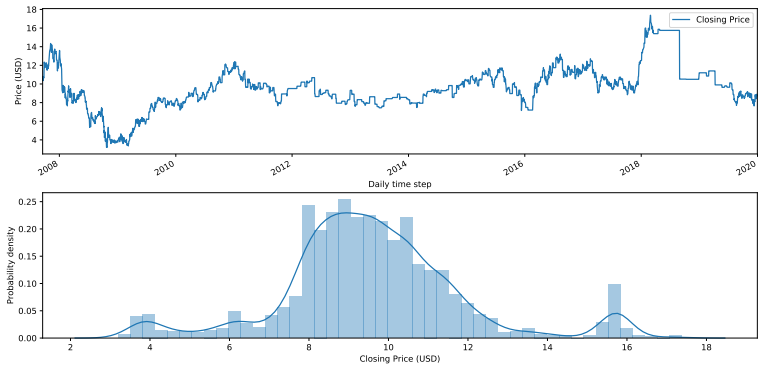

Fig. 8: Top - Composite Johannesburg Stock Exchange (JSE) closing price dataset. Bottom - Probalility distribution of the JSE dataset.

Table 6: 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
Trend lines feature size	2	2	2	2
Raw local data + Trend line feature size	21	102	6	4
Number of data instances	42279	4418	10014	1001

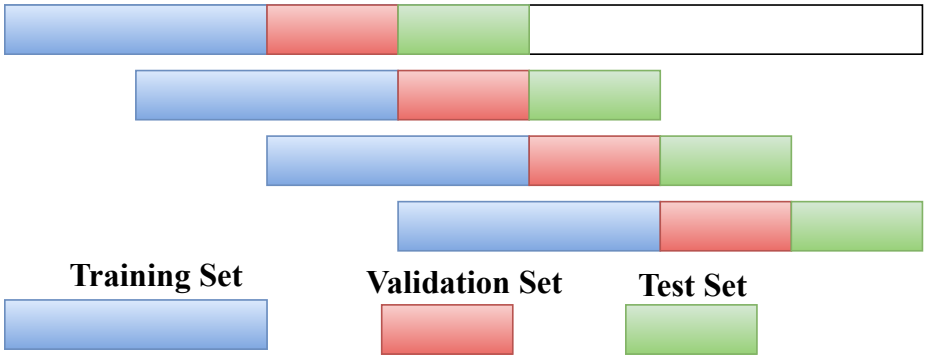

Fig. 9: An example of successive training validation test sets in overlapping partition

method, the number of splits is determined using the equation 4,

$$Number\ of\ splits = S = \lfloor \frac{T_{total\ test}}{T_{test}} \rfloor = \lfloor \frac{\alpha \times T_{total}}{T_{test}} \rfloor \quad (4)$$

where $T_{test} \rightarrow$ *the test set size*, $T_{total\ test} \rightarrow$ *the size of the union of all the test sets*, $T_{total} \rightarrow$ *the total number of data instances*, and $\alpha \in (0, 1)$. α represents the percentage of the total data instances used for hold-out evaluation, that is $T_{total\ test}$.
Given T_{total} and α, the maximum number of splits S is obtained when $T_{test} = 1$; and the minimum S when $T_{test} = \lfloor \alpha \times T_{total} \rfloor$. The maximum number of splits may be optimal, because the model will be specifically trained for each held-out data instance. However, to reduce the computation time, the test set size T_{test} is chosen from the interval $[1,\ \alpha \times T_{total})$. The greater the number of data instances T_{total}, the greater the test set size T_{test}.
Given T_{test}, α and T_{total}, obtained from the preprocessing step, the number of splits for each dataset is computed using the equation 4.
Furthermore, the validation set is made equal to the test set so that the validation error is representative of the hold-out error during the hyperparameter optimisation.
Finally, given the number of splits, the test and validation set sizes, the maximum training set size is given by the equation 5,

$$T_{training_{max}} = \lfloor T_{total} - T_{validation} - S \times T_{test} \rfloor = \lfloor T_{total} - (S + 1) \times T_{test} \rfloor \quad (5)$$

where $T_{training} \rightarrow Training\ set\ size$, and $T_{validation} \rightarrow validation\ set\ size$.
The training sets are used for model training; the validation sets for model development; and the test sets for model generalisation estimation.

Table 7: 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
Maximum training set size	4233	3968	4008	899
Used training set size	4227	3967	4008	899

7.3 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 algorihms. 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 optimal 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 optimal 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 equation 8. The equation 8 is derived from equation 6 and equation 7.

$$E' = E + E \times (S - 1) \times \omega \tag{6}$$

$$E' = E \times (1 + (S - 1) \times \omega) \tag{7}$$

$$\textit{speed-up} = \frac{E}{E'} = \frac{S}{1 + (S - 1) \times \omega} \tag{8}$$

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.*

Table 8: Our Optimal 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. [17]	0.5	5e-4	?	[600]	[32, 32]	*from S*	?	?	*N/A*

7.4 Analysis of model update with warm start:

Table 10 shows the speed-up gained with the model update with warm-start using equation 8. The RMSE achieved by the hybrid neural network with and without warm-start are also shown.
With the model update with warm-start, the training process is faster without compromising the generalisation ability of the neural network on all four datasets. The higher the number of splits, the higher the speed-up gained. The lower the warm-start fraction, the higher the speed-up gained.

Table 9: Optimal RF and GBM hyperparameters found by manual experimentation.

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 10: The reduction factor in total number of epochs (speed-up), and the average (slope and duration) RMSE using model update with/without warm-start

	Voltage	Methane	NYSE	JSE
Number of splits	8	44	5	101
Warm-start fraction	0.2	0.1	0.5	0.05
Speed-up	3.33	8.30	1.67	16.83
RMSE with warm-start	**35.79 ± 0.02**	**40.58 ± 1.43**	**44.20 ± 0.41**	**16.40 ± 0.09**
RMSE without warm-start	35.81 ± 0.01	46.49 ± 2.18	44.82 ± 0.25	17.07 ± 0.06

Table 11: Performance of MLP with three different feature types.

	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	$\mathbf{9.03 \pm 0.06}$	62.81 ± 0.04	$\mathbf{35.92 \pm 0.05}$	$\mathbf{14.56 \pm 0.19}$	$\mathbf{34.46 \pm 2.79}$	$\mathbf{24.51 \pm 1.49}$
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	90.76 ± 4.43	33.08 ± 42.08	61.92 ± 23.26	$\mathbf{19.87 \pm 0.01}$	$\mathbf{12.51 \pm 0.09}$	$\mathbf{16.19 \pm 0.05}$
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

(a) MLP

	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	10.30 ± 0.0	62.87 ± 0.0	36.59 ± 0.0	$\mathbf{14.21 \pm 0.19}$	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	$\mathbf{48.03 \pm 5.74}$	$\mathbf{31.40 \pm 3.13}$
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	86.56 ± 0.01	$\mathbf{0.41 \pm 0.08}$	$\mathbf{43.49 \pm 0.05}$	$\mathbf{19.83 \pm 0.01}$	$\mathbf{12.68 \pm 0.01}$	$\mathbf{16.26 \pm 0.01}$
Raw data + Trend lines	$\mathbf{86.50 \pm 0.01}$	0.47 ± 0.03	$\mathbf{43.49 \pm 0.02}$	20.16 ± 0.03	12.74 ± 0.02	16.45 ± 0.03

(b) LSTM

	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	$\mathbf{9.24 \pm 0.10}$	$\mathbf{62.40 \pm 0.13}$	$\mathbf{35.82 \pm 0.12}$	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	$\mathbf{37.92 \pm 4.11}$	$\mathbf{26.53 \pm 2.20}$
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Raw data	89.31 ± 1.38	12.21 ± 12.17	50.76 ± 6.78	$\mathbf{19.90 \pm 0.06}$	$\mathbf{12.48 \pm 0.21}$	$\mathbf{16.19 \pm 0.14}$
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

(c) CNN

Table 12: Comparison of RMSE error achieved by the automated algorithm selection and hyperparameter optimisation (CASH), and the best manually tuned vanilla DL model (M-DL).

	Voltage			Methane		
	Slope	Duration	Average	Slope	Duration	Average
Best M-DL	9.24 ± 0.10	$\mathbf{62.40 \pm 0.13}$	$\mathbf{35.82 \pm 0.12}$	$\mathbf{14.01 \pm 0.21}$	$\mathbf{40.09 \pm 6.95}$	$\mathbf{27.05 \pm 3.58}$
CASH	9.70 ± 0.44	62.97 ± 0.14	36.34 ± 0.29	14.93 ± 0.19	46.11 ± 4.79	30.52 ± 2.49
% improvement	-4.98	-0.91	-1.45	-2.47	**7.98**	**5.16**
	NYSE			JSE		
	Slope	Duration	Average	Slope	Duration	Average
Best M-DL	$\mathbf{86.56 \pm 0.01}$	$\mathbf{0.41 \pm 0.08}$	$\mathbf{43.49 \pm 0.05}$	$\mathbf{19.87 \pm 0.01}$	12.51 ± 0.09	$\mathbf{16.19 \pm 0.05}$
CASH	86.61 ± 0.03	0.55 ± 0.15	43.58 ± 0.09	20.00 ± 0.13	12.46 ± 0.18	16.23 ± 0.16
% improvement	-0.06	-34.15	-0.21	-0.65	**0.40**	-0.25