A comparative analysis of state-of-the-art time series forecasting algorithms

Cathal Murray
School of Computing, Eng & Intel, Sys
Ulster University
Londonderry, UK
Orchid id: 0000-0002-7729-2537

Priyanka Chaurasia
School of Computing, Eng & Intel, Sys
Ulster University
Londonderry, UK
Orchid id: 0000-0003-4249-3678

Lynsey Hollywood
Department of Hospitality & Tourism Mgmt
Ulster University
Belfast UK
Orchid id: 0000-0003-3918-6255

Damien Coyle
School of Computing, Eng & Intel, Sys
Ulster University
Londonderry UK
Orchid id: 0000-0002-4739-1040

Submission: Full/Regular Research Paper, **Research Track:** CSCI-RTAI

***Abstract*—In recent years many new algorithms have been developed for applications in speech and image processing which can be repurposed for time series prediction. This paper presents a comprehensive comparative analysis of time series forecasting capabilities of eight such state-of-the-art algorithms – namely: Vanilla Long Short-Term Memory(V-LSTM) Gated Recurrent Unit (GRU), Bidirectional LSTM(BD-LSTM), Auto encoder (AE-LSTM), Convolutional Neural Network LSTM(CNN-LSTM), LSTM with convolutional encoder (ConvLSTM), Attention mechanism networks and the Transformer network. Model performances across five different benchmark datasets including fields of interests such as finance, weather and sales are evaluated. Whether direct or iterative prediction methods are optimal for forecasting is investigated. For efficient model optimization, the asynchronous successive halving algorithm (ASHA) is applied in the training folds in a 10 *k*-fold cross validation framework. Statistical tests are used to comprehensively compare algorithm performances within and across datasets. We show that whilst there are differences between all models, the differences are insignificant for the top performing models which include the Transformer, Attention, V-LSTM, CNN-LSTM and CV-LSTM. However, the transformer model consistently produces the lowest prediction error. We also show that the iterative multistep ahead prediction method is optimal for long range prediction.**

Keywords—time series prediction, forecasting, multi-horizon, attention, transformer, LSTM

I. INTRODUCTION

Time series forecasting capability is in high demand across many challenging domains and data analytics problems. Time series forecasting involves using a model to interpret a previous sequence of values along equally spaced points in time to determine consequent values succeeding that sequence along the same given time axis. The earliest adopted methods were pioneered by Holt-Winters [1] and subsequently Box and Jenkins [2]. Since then, statistical models such as Autoregressive Integrated Moving Average (ARIMA), X11ARIMA, X12ARIMA[3][4], Seasonal Autoregressive Integrated Moving Average (SARIMA)[2], Seasonal Autoregressive Integrated Moving Average with Exogenous factors (SARIMAX), Support Vector Machine (SVM)[5], were applied to the time series forecasting domain. A limitation of such models is their inability to discern exogenous input features [6]. However, the primary constraint is a fundamental pre-established linear basis in Autoregressive Moving Average (ARMA) based models which poses difficulties for these models to learn and predict non-linear dynamics of a given time series [7]. Great improvements have been achieved by approaches that are nonlinear machine learning solutions such as kernel methods [8] gaussian processes [9] and ensemble methods[10]. However, such methods have the limitation of using a predefined nonlinear form and are unable to assimilate true nonlinear relationships. Neural networks employ supervised training and are considered universal approximators that can learn the underlying form of a data generating source. The Multi-Layer Perceptron (MLP) was one of the first neural networks used for time series forecasting and later recurrent networks have made improvements in this pursuit [11]. However, the advent of the LSTM has facilitated information from long term dependencies to make more accurate predictions [12]. Fuzzy approaches and fuzzy neural network approaches have also been employed [13][14][15][16]. In recent years, there have been developments in deep learning(DL)[17] approaches across a range of other domains, which have given rise to algorithms that have found applications in time series forecasting. For example, those focused on natural language processing and translation applications with attention-based architectures [18] being the state-of-the-art in this domain. Consequently, as such algorithms inherently perform sequence prediction, they may be strong candidates for time series prediction [19][20]. In this paper we briefly describe and thoroughly compare a range of the most recent deep learning architectures including the Vanilla Long Short Term Memory(V-LSTM)[21] which is based on the Long Short Term Memory(LSTM)[21]unit, the Gated Recurrent Unit (GRU)[22], Bidirectional LSTM(BD-LSTM)[23], Auto encoder (AE-LSTM)[24], Convolutional Neural Networks combined with LSTM(CNN-LSTM)[25], Attention mechanism network[26] and the Transformer network[27]. We also include the multilayer Perceptron(MLP)[28] network as a basic benchmark. In time series forecasting model evaluations, comparisons of one model against another are often limited due dataset limitations and varying data complexity as well as variation on the prediction horizon [29]. For example, one algorithm may be easier to train and/or perform better than another for particular data dynamic or prediction horizon and therefore there is a need to comprehensively determine which of the latest state-of-the-art solutions are suited to specific forecasting domains etc. Here we evaluate model performances across five different benchmark datasets, each with varying levels of complexity, including sales prediction, weather, a synthetic time series, a financial closing series, and an M4 forecasting competition daily time series dataset. We also investigate, for the first time, if direct or

979-8-3503-2028-2/22/$31.00 ©2022 IEEE
DOI 10.1109/CSCI58124.2022.00082

iterative methods [30] for multi-step ahead prediction (multi-horizon forecasting) are optimal for these new breed of time series forecasting models.

Time series testing methodologies must be transparent and conform to well-established techniques [29] to ensure results are generalizable to new data. Therefore, generalized and methodologically homogenous procedures should be employed to standardize algorithm testing. Here model architectures and hyperparameters are efficiently but comprehensively tested and selected in the inner *k*-fold cross validation with an asynchronous successive halving algorithm (ASHA) framework. A range of statistical tests to determine model differences are applied.

II. Recurrent and attention-based algorithms

A Multi-Layered Perceptron (MLP) neural network, used as a benchmark in this paper, can be trained to learn the relationship between lagged and embedded samples of a time series and the following samples through an iterative learning process with sufficient data availability. Networks examined in this paper are exclusively more advanced recurrent-based architectures which have generally superseded basic recurrent networks due to better long-range memory performance. For example, the attention mechanism [26] using the softmax layer, interprets the importance or relevance of data points with neighboring points. Consequently, the softmax layer allows model weights to develop an interpretation of time series dependency, which it can use to forecast future data points. Here we provide a brief overview of the various RNN-inspired deep learning (DL) architectures in this work.

Recurrent Neural Networks and LSTM: The LSTM overcomes the problem of vanishing gradients by having linear functions with recurrent connections that are fixed to unity avoiding the reduction of back-propagation gradient values. Input and output gates regulate the flow of data to avoid conflicts of storing certain inputs, ignoring others and preventing outputs from perturbing consequent cells [21]. Consequently, long-term dependency within a time series sequence may be better interpreted by a neural network using the LSTM structure.

Bidirectional LSTM: (BD-LSTM) Presenting a sequence to an output or context layer in both directions will derive a better context by accessing both future and past data points for a specific position in a sequence. The backpropagation procedure adds complexity as updating both outputs and state neurons cannot be conducted sequentially[31]. Therefore, a data sequence is divided, then start and end tokens are appended during pre-processing.

CNN Encoder LSTM Decoder: (CNN-LSTM) This model uses an initial convolutional layer to perform feature extraction on input data. Consequently, the extracted data features are applied to an LSTM section of the model to perform sequence prediction. The first application of a CNN-LSTM was for visual activity recognition, image captioning, and video description [32]. Another form of this system involved a binary CNN encoder and a gated recurrent unit (GRU) proposed by Yoshua Bengio et al. [22]

Convolutional LSTM: (ConvLSTM) This model is strongly related to the CNN-LSTM. However, the convolutional layers are directly built into the hidden layers of each LSTM cell, and the structure does not contain a strict enveloping encoder-decoder structure [33].

Autoencoder: (AE-LSTM) An Autoencoder generally consists of an encoder that maps inputs from a high dimensionality to a low dimensional vector and a decoder that reconstructs the input from the low dimensional vector to a higher dimensional space. One pioneering application for this structure was its use in language translation [34]. Training the autoencoder involves applying sentences from one language as input to an encoder network with a numerical representation as a target. This target, in turn, is used as training input to another neural network system with sentences from another language as target data. Forecasting one word following another using conditional probability was introduced by Neco and Forcada [35] and from this prediction aspect, the autoencoder is shown to be capable of time series forecasting. The structural arrangement of the autoencoder used for this paper consists of having LSTM network layers for both encoder and decoder, thus creating the ability to stack autoencoder units more effectively. Sutkever et al. [24] used this approach to build NLP translation machines. Here we refer to this structure as the Autoencoder LSTM (AE-LSTM).

Attention mechanism: One restriction of the auto-encoder was the fixed length vector between the encoder and decoder. Benjio et al. [26] introduced the attention mechanism as a solution. Instead of forming a context vector of the last state of the encoder, the output of the decoder is forced to depend on a weighted combination of all input states. The attention mechanism may perform feature capture of a time series due to its capacity to focus more on some intervals than others.

Transformer: Aswani et al. [27] has developed a natural language translation model known as a transformer. The transformer depends heavily on self-attention layers and excludes any recurrent or convolutional structures within its architecture. The Transformer architecture is a complex encoder-decoder structure that, when applied in NLP applications, consists of different multi-head attention layers to compare sentences with themselves and target sentences from another language. The transformer processes whole sentences and does not rely on hidden states to interpret the dependencies of previous words. Thus, it eliminates the process of losing previous information within a sequence, which LSTM networks use as part of their function. The forget gate inside an LSTM unit is designed to remove irrelevant and non-influential information from the state cell where further predictions no longer depend upon that information.

III. Methodology

A. Datasets

Models may vary in their ability to generalize to data that have not been previously 'seen' by the model during training. This may depend not only on the type of model and its' trained hyperparameters but also on the nature of the dataset e.g., non-stationary dynamics or level of complexity. Also, a model may predict meteorological time series well after training but have

poor forecasting performance on a financial time series. It is, therefore, important to compare models across a range of datasets with different dynamics.

Five datasets are selected for this paper as shown in table I. along with decomposed properties including trend strength, seasonal strength, Shannon entropy, and the number of anomaly data points.

TABLE I. DATASET PROPERTIES

Dataset	Size	Frequency	Type	Trend Strength	Seasonal Strength	Shannon Entropy
Pm2_5	430K	Hourly	Large	0.225	0.604	7.729
D3497	3400	Daily	M4 Competition	$7.3*10^{-4}$	0.689	11.902
Sales1	1800	Weekly	Shop Sales	0.3158	0.435	9.192
ARMA	N/A	N/A	Synthetic	0.082	0.689	11.966
APPL	2000	Daily	Financial	3.3*10-3	0.730	11.918

B. Mutli-step ahead prediction approaches

The prediction horizon is the length of time into the future for which predictions are to be made. Here we compare the iterative and direct forecasting methods, for multi-horizon prediction.

Iterative Forecasting: The iterative forecasting technique used in this analysis is the rolling origin evaluation [29] procedure where the models are trained to complete one step ahead prediction. Repeated one step ahead predictions within the forecast horizon are used as model inputs until the iteration reaches the required prediction horizon. Fig. 1. shows the iterative prediction pseudocode.

Direct forecasting: Direct forecasting involves directly training the model to predict the time series at the required sample in the future i.e., the forecast sequence is predicted in a one-shot fashion. For multiple horizon direct forecasting, a separate model is created for each forecast horizon. Madrikas et al. [36] analyse prediction horizons up to 18 steps ahead (t+18). In this analysis, we analyse t + 1 and t + 18 prediction horizons for all models/datasets.

Multi-step iterative prediction pseudocode

history $\leftarrow (X_t, X_{t-1}, X_{t-2}, \dots, X_{t-n})$;
prediction array = []
prediction horizon $\leftarrow P$
for $i \leftarrow 1$ to P **do**
  input $\leftarrow$ history
  enter input into Model to get $\hat{y}$
  history $\leftarrow (\hat{y}_t, X_t, X_{t-1}, X_{t-2}, \dots, X_{t-n+1})$
  prediction array $\leftarrow \hat{y}_t$
end
output prediction array $\leftarrow (\hat{y}_1, \hat{y}_2, \hat{y}_3, \dots, \hat{y}_p)$

Fig. 1. Basic Multi-Horizon iterative prediction code

IV. EVALUATION METHODS

A. Mean absolute prediction error

The Mean Absolute Scaled Error (MASE)[37] measurement is chosen to evaluate the performance of each network for each dataset as it is a metric that is independent of the scale of the data and thus can be used to compare various model/dataset combinations. The Mean Absolute Error (MAE)[37] measurement, when applied to time series forecasting, is the calculation of the mean absolute value of the pairwise difference between the prediction from a model and the actual time series prior to forecasting.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} abs(\hat{y}_i - y_i) \quad (2)$$

The naïve MAE uses the prior time series as a forecast and uses all time points before the final data point as the prior to that time series.

$$MAE_{naive} = \frac{1}{N-1}\sum_{i=2}^{N} abs(y_i - y_{i-1}) \quad (3)$$

MASE is the ratio of MAE and naïve MAE

$$MASE = \frac{MAE}{MAE_{naive}} \quad (4)$$

B. k-fold cross validation

k-fold cross-validation is a recommended solution for hyperparameter optimization assessment [29] to ensure generalizable and trustworthy results. *k*-fold cross-validation has the advantage of reducing measurement variance when estimating model performance [38]. It is also necessary to ensure no data leakage between training and testing so therefore all hyperparameters should be selected by evaluating performance on the training data of the *k*-fold CV. This can be done by inner-outer fold cross validation but is time consuming. Here instead, for each iteration of the *k*-fold analysis, a hyperparameter tuning method known as the Asynchronous Successive Halving Algorithm (ASHA)[39] is applied using only the training data within each *k*-fold iteration. Fig. 2. shows the pseudocode for the complete *k*-fold CV and the ASHA approach is described below.

K fold Cross Validation with ASHA hyperparameter tuning

1. Define a set of hyperparameter combinations, *C*, for the current model
2. Divide data into *K*+1 folds of equal size. For time series prediction, do not shuffle folds
3. For fold k_i within *K*+1 folds:
   (a) Set fold k_{i+1} as the test set
   (b) Set folds k_0 to k_i as the train set
   (c) For parameter combination *c* in C:
      (i) Perform ASHA hyperparameter optimization using the train set
      (ii) ASHA process selects the best trained model and evaluates its performance
4. *K* best models are collected, each one specifically trained for each *k*-fold
5. Evaluate each model using its corresponding test data set, which is fold k_{i+1}, employing a prediction method to producing a performance value
6. Calculate average performance over *k*-folds

Fig. 2. Pseudocode for *k*-fold cross-validation with ASHA hyperparameter tuning

C. ASHA Hyperparameter Tuning

Many hyperparameter tuning algorithms have been proposed in recent years based on the multi-armed bandit(MAB) problem (Katehakis and Veinott, 1987)[40] where a fixed set of resources are distributed among a set of choices to produce maximum gain. One approach to optimizing hyperparameters using MAB is that of the sequential halving algorithm(SHA)[39] where a given budget is split evenly across a set number of elimination rounds. Bandit machine arms are tried. At the end of each round, a specific number of arms are ruled out, and the rest are selected for the next round. This procedure is repeated until a single bandit machine is discovered for optimum gain. Applying SHA to hyperparametric optimization [41] involves identifying

hyperparameter configurations as bandit arms. The budget is usually the number of epochs used to train a model, and a models' error or accuracy is treated as the gain. The SHA algorithm consists of an outer loop structure where each loop is referred to as a rung. A fixed ratio known as the reduction factor determines the amount of budget/epochs as well as the number of arms/hyperparameter combinations being processed within each rung. The number of hyperparameters is inversely proportional to the number of epochs used within each rung. For each rung, a number of hyperparameter configurations/trials are retained for the following rung based on their performance, and the remaining are discarded. Finally, a single hyperparameter set is selected with the most promising performance. The drawback with this algorithm is the lagging issue and slow trials as the completion of a rung require waiting for all training with various hyperparameters to finish their epoch number before commencing the next rung. The solution to this bottleneck was to introduce parallelism using the Asynchronous Sequential Halving Algorithm (ASHA)[39] which runs different rungs concurrently and fills successive rungs with higher-performing hyperparameter configurations as and when they are ready. In this analysis, the number of trials applied to each model/dataset/*k*-fold combination is 28.

Asynchronous Successive Halving (ASHA)

input minimum resource r, maximum resource R, reduction factor n, minimum early-stopping rate s
function ASHA()
  repeat
    for each free worker **do**
      $(\theta; k)$ = get job()
      run then return val loss($\theta; rn^{s+k}$)
    end for
    for completed job $(\theta; k)$ with loss l **do**
      Update configuration θ in rung k with loss l
    end for
  until desired
end function
function get job()
  // Check if there is a promotable config.
  for $k = |\log_n (R/r)| - s - 1, \ldots, 1, 0$ **do**
    candidates = top_k(rung k; | rung k|/n)
    promotable = { $t \varepsilon$ candidates : t not promoted }
    | promotable| > 0 **then**
      return promotable[0], $k + 1$
    end if
      // If not, grow bottom rung.
      Draw random configuration θ.
    return θ,0
  end for
end function

Fig.3 Asynchronous successive halving pseudocode

D. Computing architecture for analysis

For 5 datasets and 9 models with 10-fold cross validation there are 450 jobs required on the distribution cluster for one prediction horizon. Each job is assigned 8 CPUs and a single NVIDIA V100 GPU to train a model.

V. Results

Model acronyms for all tables and figures in this test results section are abbreviated. V-LSTM -> V, BD-LSTM -> BD, CNN-LSTM -> CNN, ConvLSTM -> CV, AE-LSTM -> ED (Encoder-Decoder), Attention -> Atten, Transformer -> Tranf. Reference to prediction horizons will be prefaced by 'PH' followed by the interval number, e.g. next step and 18 step prediction are denoted as PH1 and PH18.

A. Single and Multi-Horizon Direct Method

Fig. 3. (a) and (b) show typical error distribution boxplots for most datasets using the direct prediction method. The pm2_5 dataset boxplots shown are similar to boxplots of all other datasets. Fig. 3. (a) and (b) reveal a significant error distribution difference between MLP ($p<0.05$) and the rest of the models. However, the GRU, BD-LSTM and ED-LSTM in PH1 pm2_5 show larger distribution spreads compared to optimum models. For instance, the Games Howell test comparison between CNN-LSTM and BD-LSTM for the PH1 pm2_5 reveals a significant difference $T(8,80)=-7.40$, $p < 0.05$. Fig. 3. (c) and (d) show the boxplot distributions of the top-performing models for PH18 ARIMA and pm2_5 and reveal that the model error distributions are close to one another except for the Attention model which performs poorly in comparison to other optimum models for the direct prediction approach. Statistically, the Games Howell comparison test for distributions CNN-LSTM and Attention for PH18 pm2_5 reveals a significance $T(4,40)=-3.26$, $p < 0.05$. However, the model distributions of the optimum models across the datasets D3497, Sales1 and APPL datasets reveal no significant difference between their optimum model distributions. To determine the best of these models in general the average over all test folds of the k-fold CV for each model/dataset combination are sorted from minimum error score to maximum and ranked in that order. Table II. Shows the top 3 ranked for each dataset.

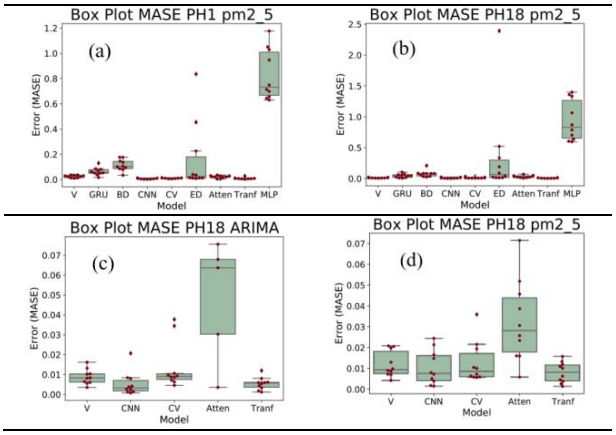

Fig. 3. Box plots of Direct Multi-Horizon PH1 vs. PH18 distributions of all models for given dataset. (a) PH1 for pm2_5 dataset and all models (b) PH18 for pm2_5 dataset and all models. (c) optimum model PH18 for PM2_5 dataset (d) optimum model PH18 for PM2_5 dataset

TABLE II. Direct Method Best Performing Models

Direct Multi-Horizon Best Models					
Rank(PH)	***pm2_5***	***Sales1***	***D3497***	***ARMA***	***APPL***
1st (PH1)	CNN	Tranf	Tranf	Tranf	GRU
2nd (PH1)	Tranf	CNN	V	CNN	CNN
3rd (PH3)	CV	V	CNN	V	CV
1st (PH18)	Tranf	Tranf	V	CNN	BD
2nd (PH18)	CNN	CNN	Tranf	Tranf	V
3rd (PH18)	V	V	CV	BD	CNN

Here we can see that the Transformer, CNN-LSTM, CV-LSTM, and V-LSTM models are consistently ranked in the top 3 for the majority of datasets. However, the financial closing datasets best suit the GRU and V-LSTM networks. Even though, in most cases, there is no statistically significant difference between the top 3 performing models. Transformers have the lowest error in 5/10 datasets/prediction horizons. In contrast, no other algorithm performance has the lowest error more than once except CNN 2/10, so the results suggest that the Transformer is the most versatile and accurate model. Transformers are the most recent approach to DL and are driving the state-of-the-art speech and text processing algorithms.

B. Single and Multi-Horizon Iterative Method

Fig. 4. (a) and (b) present the pm2_5 dataset boxplot for PH1 and PH18 and show a typical error distribution boxplot pattern for most datasets using the iterative prediction method. All models reveal a significant difference from the MLP benchmark error distribution $p > 0.05$ for all other datasets tested. Fig. 4. (a) and (b) show the AE-LSTM and BD-LSTM error distributions are larger than the optimum model distributions. They are also statistically significantly different from the optimum model distributions $p > 0.05$. For, example, The Games Howell post hoc test between CNN-LSTM and BD-LSTM revealed a significant difference for PH1 pm2_5 $T(8,81)$=-8.96, $p < 0.05$ and PH18 pm2_5 dataset $T(8,81)$=-3.89, $p < 0.05$. Fig. 4. (c) shows that the model distributions for the optimum models, when compared with each other, show no significant difference and this is true for all datasets using the iterative method. Fig 5. (d) shows the average CV error for all models over all datasets. This boxplot shows a larger distribution spread for the BD-LSTM and AE-LSTM relative to non-benchmark models under test. For multiple horizons, the best-performing models infrequently change, as shown in table III. Moreover, the best candidates for most model/dataset combinations are ATTENTION MECHANISM, CNN-LSTM, CV-LSTM, V-LSTM, and the TRANSFORMER.

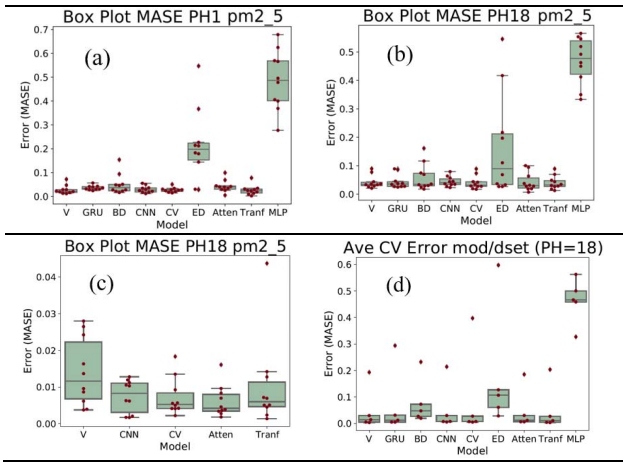

Fig. 4. (a) and (b) show the pm2_5 PH1 and PH18 boxplots. (c) optimum model PH18 for PM2_5 dataset (d) The average CV error over all models and datasets boxplot.

TABLE III. ITERATIVE METHOD BEST PERFORMING MODELS

Iterative Multi-Horizon Best Models					
Rank(PH)	***pm2_5***	***Sales1***	***D3497***	***ARMA***	***APPL***
1st (PH1)	CNN	Tranf	Tranf	Tranf	Atten
2nd (PH1)	Tranf	CNN	CV	CNN	V
3rd (PH1)	CV	V	V	V	Tranf
1st (PH18)	Atten	Tranf	Tranf	V	Atten
2nd(PH18)	CV	GRU	CV	Tranf	V
3rd (PH18)	CNN	V	V	GRU	Tranf

The exception is for the Sales1 dataset for which the GRU model as the second-best performance.

C. Correlation over all datasets with dataset attributes.

The correlation between model error scores for each dataset and the dataset attributes to determine if any model is particularly suited to data with specific properties. Table IV. presents correlation results for PH1 and PH18 horizons for both direct and iterative methods. The following observations are made. Trend strength is negatively correlated with error across all prediction methods and prediction horizons while the seasonal strength and Shannon entropy are positively correlated. The optimum models consistently have medium correlation strength regardless of prediction horizon and prediction method with little difference between models.

TABLE IV. CORRELATION OF MODEL ERROR AND DATASET ATTRIBUTES

	V	GRU	BD	CNN	CV	ED	Atten	Tranf	MLP
Direct PH1									
TS	-0.426	-0.566	-0.778	-0.427	-0.456	-0.739	-0.455	-0.428	-0.505
SS	0.42	0.572	0.669	0.405	0.422	0.634	0.441	0.409	0.662
SE	0.329	0.389	0.539	0.361	0.389	0.55	0.377	0.352	-0.022
Direct PH18									
TS	-0.44	-0.407	-0.167	-0.416	-0.449	-0.463	-0.477	-0.432	-0.301
SS	0.425	0.413	0.189	0.396	0.43	0.487	0.468	0.413	0.368
SE	0.361	0.299	0.042	0.347	0.374	0.316	0.406	0.359	0.06
Iterative PH1									
TS	-0.403	-0.573	-0.584	-0.406	-0.454	-0.562	-0.367	-0.407	-0.412
SS	0.396	0.579	0.556	0.38	0.435	0.555	0.36	0.388	0.572
SE	0.312	0.389	0.436	0.35	0.373	0.435	0.301	0.335	0.412
Iterative PH18									
TS	-0.382	-0.394	-0.456	-0.406	-0.417	-0.285	-0.385	-0.403	0.194
SS	0.363	0.372	0.456	0.38	0.39	0.374	0.359	0.383	0
SE	0.311	0.328	0.348	0.35	0.354	0.147	0.343	0.333	-0.242

[a] [b] TS = Trend Strength, SS = Seasonal Strength, SE = Shannon Entropy

Concerning the direct method, all correlation strengths for the bidirectional model are significantly reduced for a larger prediction horizon of PH18 but this reduction is not so pronounced in the iterative method case. When considering the iterative case, the AE-LSTM and the MLP model have sharp reductions in attribute strengths from prediction horizon PH1 to PH18.

D. Direct vs Iterative method

There are slight differences observed for PH1 direct and iterative method even though it would be assumed that identical results would be observed as no iteration occurs for one-step ahead prediction however, for iterative methods, a portion of training is appended to the test data to alleviate a sharp increase in error. Therefore, datasets D3497 and APPL have different top 3 models for PH1 horizons for direct and iterative methods but the difference in error rates are insignificant. A pairwise t-test was also performed on PH18 iterative vs. PH18 direct method model/dataset error groups. Eleven significant differences were

discovered from 44 comparisons. Besides the significant differences for MLP, there are significant differences for the following dataset/model combinations: pm2_5/GRU, pm2_5/Attention, Sales1/AE-LSTM, ARIMA/V-LSTM, ARIMA/GRU, ARIMA/ConvLSTM, ARIMA/Transformer, APPL/BD-LSTM.

Fig. 6. reveals that the iterative method produces lower error scores than the direct method in all cases for PH18.

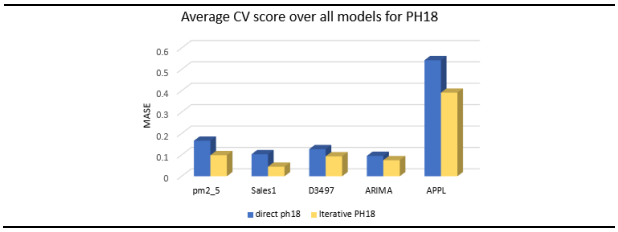

Fig. 6. Average CV score over all models. Direct at PH18 and Iterative at PH18

E. Training Time

Table IV. shows the average time taken for the training of each model over the pm2_5 datasets. The convolutional models can be trained the fastest and the convolutional LSTM model takes the least time to train, with an average training time of under 25 minutes to produce a model.

TABLE V. TRAINING TIME

Model	Conv	CNN	GRU	V	BD	Tranf	Atten	AE	MLP
Time (hh:mm)	00:24	01:05	01:18	2:20	03:59	06:56	10:17	10:22	10:46

VI. DISCUSSION

Model error distributions of models for each dataset as well as error distributions across all datasets were examined. When examining error distributions, a narrow distribution indicates greater performance consistency. Moreover, a lower error distribution mean is an indication of better performance. Throughout all comparisons, the models Bi-LSTM, AE-LSTM and GRU were proven to perform poorly in contrast to the rest of the state-of the-art models under test. There have been many investigations and attempts to discover the primacy of the direct versus the iterative strategy using neural networks. Mamat and Samad [42] found the superiority of the direct approach for long-distance predictions. Hamzacßebi et al. [30] also have found the direct method to give better results. Boné R. and Crucianu M. [43] suggest that direct method would be superior to indirect method as the error from prediction created at each iteration accumulates for further iterations in the iterative method. However, the empirical evidence of this analysis does not support this theory using these models. The iterative method produces lower error predictions and has proven to be the optimal prediction method. The correlation analysis examined the correlation between model error score and dataset attributes. Here we see that suboptimal models such as the Bidirectional, Autoencoder and MLP network lose correlation strengths dramatically depending on the prediction method.

VII. CONCLUSION

This paper comprehensively compares many of the latest neural network architectures that can be deployed for time series forecasting. Rigor was maintained using cross-validation techniques that employed ASHA hyperparameter tuning for all models ensuring identical test and conditions for all models with each receiving fair treatment. Five different datasets with varying characteristics are used for evaluation. Whilst there are differences between all models, the differences are insignificant for the top performing models which include the Transformer, Attention, V-LSTM, CNN-LSTM, and CV-LSTM. These models repeatedly reveal themselves as the optimum models for overall prediction methods and most datasets. Ranking based purely on MASE average values reveals that the Transformer is the most consistent and reliable high-performance model with the greatest number of best performance results shown over all the tested datasets as shown in table II and III. The transformer consists of a structure which can view the entire sequence using a concentrated level of attention over the data to seek relevant importance information with regard to sequence prediction without having to remove or ignore prior sequence data point information. This analysis has also shown the iterative method is best achieving significantly better performance and produced lower on average error rates across all datasets and models. The correlation analysis reveals there is little difference between optimum models when correlated with dataset attributes. In summary, to the best of the authors' knowledge this is the first comprehensive comparative analysis of state-of the-art deep learning approaches applied to time series prediction, some of which have been extensively and successfully applied in other domains. The paper serves as a guide to other researchers on which model are best for time series prediction. Future work will involve extending the analysis to additional datasets with varying dynamic and characteristics as well as greater prediction horizons.

VIII. ACKNOWLEDGEMENT

We are grateful for access to the Tier 2 High Performance Computing resources provided by the Northern Ireland High-Performance Computing (NI-HPC) facility funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant Nos. EP/T022175/ and EP/W03204X/1. DC is grateful for the UKRI Turing AI Fellowship 2021-2025 funded by the EPSRC (grant number EP/V025724/1). CM is supported by a Department for Economy Northern Ireland funded PhD Scholarship.

IX. REFERENCES

[1] P. R. Winters, "Forecasting Sales by Exponentially Weighted Moving Averages," *Manage. Sci.*, vol. 6, no. 3, pp. 324–342, Apr. 1960.

[2] I. M. Chakravarti, G. E. P. Box, and G. M. Jenkins, "Time Series Analysis Forecasting and Control.," *J. Am. Stat. Assoc.*, vol. 68, no. 342, p. 493, Jun. 1973.

[3] J. Shiskin, A. H. Young, and J. C. Musgrave, "The X-11 variant of the census method II seasonal adjustment program." U.S. Department of Commerce, 1967.

[4] E. B. Dagum, "A new method to reduce unwated ripples and revisions in trend-cycle estimates from X-11-ARIMA," *Survey Methodology*, vol. 22, no. 1. pp. 77–83, 1996.

[5] H. Yang, K. Huang, I. King, and M. R. Lyu, "Localized support vector regression for time series prediction," *Neurocomputing*, vol. 72, no. 10–12, pp. 2659–2669, Jun. 2009.

[6] T. Chen *et al.*, "TADA: Trend Alignment with Dual-Attention Multi-task Recurrent Neural Networks for Sales Prediction," *Proc. - IEEE*

Int. Conf. Data Mining, ICDM, vol. 2018-Novem, pp. 49–58, 2018.
[7] V. . & O. T. Haggan, "Amplitude-Dependent Autoregressive Time Series Model," *Biometrika*, vol. 68, no. 1, pp. 189–196, 1981.
[8] S. Chen, X. X. Wang, and C. J. Harris, "NARX-based nonlinear system identification using orthogonal least squares basis hunting," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 1, pp. 78–84, 2008.
[9] R. Frigola and C. E. Rasmussen, "Integrated pre-processing for Bayesian nonlinear system identification with Gaussian processes," in *52nd IEEE Conference on Decision and Control*, 2013, no. 3, pp. 5371–5376.
[10] A. Bouchachia and S. Bouchachia, "Ensemble Learning for Time Series Prediction," in *First International workshop on Nonlinear Dynamics and Synchronization*, 2008, no. January.
[11] R. Khalil, "Comparison of Four Neural Network Learning Methods Based on Genetic Algorithm for Non-linear Dynamic Systems Identification," *Alrafidain.Engineering-Coll-Mosul. …*, vol. 20, no. 1, pp. 8–12, 2011.
[12] J. G. Taylor, "Univariate and Multivariate Time Series Predictions," no. January, pp. 11–22, 2002.
[13] D. Coyle, G. Prasad, and M. McGinnity, "Faster self-organizing fuzzy neural network training and improved autonomy with time-delayed synapses for locally recurrent learning," *Syst. Circuit Des. Biol. Intell. Learn.*, pp. 156–183, 2010.
[14] D. Coyle, G. Prasad, and T. M. McGinnity, "Faster self-organizing fuzzy neural network training and a hyperparameter analysis for a brain-computer interface," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 39, no. 6, pp. 1458–1471, 2009.
[15] D. Coyle, G. Prasad, and T. M. McGinnity, "On utilizing self-organizing fuzzy neural networks for financial forecasts in the NN5 forecasting competition," *Proc. Int. Jt. Conf. Neural Networks*, pp. 18–23, 2010.
[16] G. Prasad, G. Leng, T. McGinnity, and D. Coyle, "On-line Identification of Self-organizing Fuzzy Neural Networks for Modelling Time-varying Complex Systems," in *Evolving Intelligent Systems: Methodology and Applications, IEEE Press Series on Computational Intelligence*, P. Angelov, D. Filev, and N. Kasabov, Eds. John Wiley & Sons, 2010, pp. 256–296.
[17] Y. Bengio, *Deep Learning Architectures for AI*. 2009.
[18] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," pp. 1–26, Oct. 2014.
[19] H. Zhou *et al.*, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," *35th AAAI Conf. Artif. Intell. AAAI 2021*, vol. 12B, pp. 11106–11115, 2021.
[20] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are Transformers Effective for Time Series Forecasting?," pp. 1–15, 2022.
[21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[22] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," Sep. 2014.
[23] A. Graves, J. Schmidhuber, L. K. Halderman, and C. Chiarello, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5–6, pp. 602–610, 2005.
[24] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112, 2014.
[25] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *34th Int. Conf. Mach. Learn. ICML 2017*, vol. 3, pp. 2029–2042, 2017.
[26] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *Conf. Int. Conf. Learn. Represent. ICLR 2015*, pp. 1–15, Sep. 2015.
[27] A. Vaswani *et al.*, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 5999–6009, Jun. 2017.
[28] O. M. Remus W., "Neural Networks for Time-Series Forecasting," *Int. Ser. Oper. Res. Manag. Sci. B. Ser. (ISOR, Vol. 30)*, vol. 30, 2001.
[29] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Inf. Sci. (Ny).*, vol. 191, pp. 192–213, 2012.
[30] C. Hamzaçebi, D. Akay, and F. Kutay, "Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting," *Expert Syst. Appl.*, vol. 36, no. 2 PART 2, pp. 3839–3844, 2009.
[31] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.
[32] J. Donahue *et al.*, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 677–691, 2017.
[33] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Adv. Neural Inf. Process. Syst.*, vol. 2015-Janua, pp. 802–810, 2015.
[34] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," *EMNLP 2013 - 2013 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, no. October, pp. 1700–1709, 2013.
[35] R. P. Neco and M. L. Forcada, "Asynchronous translations with recurrent neural nets," *IEEE Int. Conf. Neural Networks - Conf. Proc.*, vol. 4, pp. 2535–2540, 1997.
[36] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward," *PLoS One*, vol. 13, no. 3, p. e0194889, Mar. 2018.
[37] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecast.*, vol. 22, no. 4, pp. 679–688, 2006.
[38] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," *Int. Jt. Conf. Artif. Intell.*, vol. 5, no. March 2001, 1995.
[39] L. Li *et al.*, "A System for Massively Parallel Hyperparameter Tuning," 2018.
[40] M. N. Katehakis and A. F. Veinott, "Multi-Armed Bandit Problem: Decomposition and Computation.," *Math. Oper. Res.*, vol. 12, no. 2, pp. 262–268, 1987.
[41] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," *Proc. 19th Int. Conf. Artif. Intell. Stat. AISTATS 2016*, pp. 240–248, 2016.
[42] M. Mamat and S. A. Samad, "Comparison of iterative and direct approaches for multi-steps ahead time series forecasting using adaptive Hybrid-RBF neural network," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, pp. 2332–2337, 2010.
[43] R. Boné and M. Crucianu, "Multi-Step-Ahead Prediction with Neural Networks," *Eur. J. Econ. Soc. Syst.*, vol. 17, no. 1–2, pp. 85–98, 2004.