SUMMARY: Correlated Time Series Forecasting using Modular Multi-Task Deep Neural Networks

Răzvan-Gabriel Cîrstea, Darius-Valer Micu, Gabriel-Marcel Mureşan

June 7, 2018

The problem the paper explores is forecasting of time series. Though the problem is not new, we aim to provide a novel method that uses deep learning to improve upon the state of the art.

Given a set of time series $X = \langle X^{(1)}, X^{(2)}, ..., X^{(n)} \rangle$, where $X^{(i)} = \langle x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)} \rangle$, $i \leq n$ the paper aims to predict p steps ahead window, i.e. $\langle x_{a+l+1}^{(1)}, x_{a+l+2}^{(1)}, ..., x_{a+l+p}^{(1)} \rangle$.

The paper presents a novel modular neural network that exploits the Convolutional Neural Network (CNN) capabilities of capturing features and combines it with Recurrent Neural Network (RNN) in the scope of regression. During experiments the proposed model outperforms its competitors and proves itself to be useful for long and short term forecasting.

The basic intuition of both is to separate the CNN for each time series and merge the convoluted features as input for an RNN. The second model expands CNN into a full auto-encoder that reconstructs the input time series, making it a multi-task learning model. For the experiments we used mainly a real world data set that consists of measurements of chemical concentrations within an aeration tank of a sewage treatment facility, proving a good combination of industrial and biological processes.

As second data set used in testing consisted of Google Trends search data.

Evaluation of the model shows lower errors compared to established linear algorithms for time series forecasting and neural networks, models that do not take advantage of correlations in time series. Two variations of the model are described.

Correlated Time Series Forecasting using Modular Multi-Task Deep Neural Networks

Răzvan-Gabriel Cîrstea
Aalborg University
rcirst16@student.aau.dk

Darius-Valer Micu
Aalborg University
dmicu16@student.aau.dk

Gabriel-Marcel Mureşan
Aalborg University
gmures16@student.aau.dk

***Abstract*—The problem the paper explores is forecasting of time series. Though the problem is not new, we aim to provide a novel method that uses deep learning to improve upon the state of the art. The paper presents a novel modular neural network that exploits the Convolutional Neural Network (CNN) capabilities of capturing features and combines it with Recurrent Neural Network (RNN) in the scope of regression. During experiments the proposed model outperforms its competitors and proves itself to be useful for long and short term forecasting. Evaluation of the model shows lower errors compared to established linear algorithms for time series forecasting and neural networks, models that do not take advantage of correlations in time series. Two variations of the model are described. The basic intuition of both is to separate the CNN for each time series and merge the convoluted features as input for an RNN. The second model expands CNN into a full auto-encoder that reconstructs the input time series, making it a multi-task learning model. For the experiments we used mainly a real world data set that consists of measurements of chemical concentrations within an aeration tank of a sewage treatment facility, proving a good combination of industrial and biological processes.**

I. Introduction

In recent years big data is an increasingly common term, with companies collecting more and more data at constantly higher rates. From creating an advertisement profile of a user to forecasting energy requirements of a city, time dependent measurements are collected with the purpose of categorization, outlier detection and forecasting.

These sequences of measurements are called time series. Fields of industry such as stock exchange financial predictions, web traffic analysis, road traffic, biological ecosystems, and weather predictions present some sort of time dependent correlated measurements that can be modeled as time series.

For example [1] shows how time series can be used while teaching an AI to drive on a highway, which was the first important step in the self-driving car industry. Another use case for time series forecasting is the energy industry, [2] proved that using neural networks it is possible to estimate the wind speed in certain areas, resulting in more accurate energy production estimations. [3] and [4] describes how using neural networks can be beneficial to the healthcare system by automatically detecting diseases such as Alzheimer or thyroid problems.

Since in the real world many of those systems interact with each other, those interactions reflect within the measurements and can be quantified using correlation. Using sets of *correlated time series* various techniques can be used for the purpose of time analysis and forecast. They start with linear methods such as regressions and moving on to more complex non linear methods. The current state of the art being the use of neural networks [5].

The structure of the paper is as follows, chapter II offers a formal definition of a time series and introduces Convolutional and Recurrent Neural networks that are to be used in the following chapters. In II-B the problem statement of the paper is concisely presented, together with the novelties of the proposed models. Within chapter III are showcased other papers related to time series forecasting that are considered to be the state of art. From IV the proposed model is formally explained with V and VI presenting the testing scenarios and the results of the model respectively. In chapter VII the final conclusion along with considerations observed during the experiments are presented. Finally in chapter VIII we address future adjustments that could potentially improve the proposed models.

II. Preliminaries

A. Definitions

A time series $X^{(i)}$ is a sequence of measurements x ordered chronologically, i.e. $X^{(i)} = \langle x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)} \rangle$. Usually the time interval between two readings is constant. We refer to $x_t^{(i)}$ as the measurement belonging to series $X^{(i)}$ taken at time tick t. For the scope of this paper we are going to work only with time series where each measurement is a decimal number.

A set of correlated time series is denoted as X, with the assumption that there exists a relation between the measurements of the component time series. The assumption can be confirmed by computing the pairwise correlation coefficients between the dependent (first) time series, and the remaining independent time series using methods such as Spearman and Pearson. The correlations coefficients have values in the range of [-1, 1], with 0 indicating no correlation and -1 and 1 indicating a high negative and positive correlation respectively.

Table I presents all the notations used in the paper next to their respective mathematical and semantic interpretations.

Symbol	Definition
X	$X = \{X^{(1)}, X^{(2)}, ..., X^{(n)}\}$ Set of correlated time series
$X^{(i)}$	$X^{(i)} = \langle x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)} \rangle$, $i \leq n$ Sequence of all readings from time series i
$\hat{X}^{(i)}$	$\hat{X}^{(i)} = \langle \hat{x}_1^{(i)}, \hat{x}_2^{(i)}, ..., \hat{x}_m^{(i)} \rangle$, $i \leq n$ Reconstruction of $X^{(i)}$
Z	$Z = \langle z_1, z_2, ..., z_p \rangle$ Sequence of predicted values
$x_t^{(i)}$	$X^{(i)}$ at time t, $t \leq m$ Measurement at specific time t of $X^{(i)}$
$\hat{x}_t^{(i)}$	$\hat{X}^{(i)}$ at time t, $t \leq m$ Reconstructed measurement at time t
$z_t^{(i)}$	Z at time t Prediction at specific time t
m	$m = \|X^{(i)}\|$ Length of time series $X^{(i)}$
a	$a < (m - p - l)$ Current prediction window
n	$n = \|X\|$ Number of correlated time series
l	$l = \|X^{(i)}[(a+1)..(a+l)]\|$ Length of a model input at window a
p	$p = \|Z\|$ Prediction size
W	$W = \{W^1, W^2, ..., W^j\}, j > 0$ Set of weights matrices
b	$b = \{b^1, b^2, ..., b^k\}, k > 0$ Set of bias matrices
η	$\eta \in \mathbb{R}$ Learning rate

TABLE I: General notations

Neural networks are general mathematical models, inspired from biology, that automatically learn how to map a set of inputs to a set of outputs. Within the context of time series, they are also capable of automatically capturing the dynamics of time series for the purpose of classification and regression. Neural networks can be classified into two categories, feed forward and recurrent networks. The main difference between the two is how they propagate the data through time, one making use of recurrent cells while the other does not.

J. T. Connor et al. [6] proposed the recurrent neural network (RNN) model that in theory is able to capture long and shot term dependencies. Due to its recurrent nature it has been proven that such a model requires more computation power to be trained, compared with a feed forward network. In addition it suffers from the vanishing gradient problem [7] that further makes the network difficult to train.

B. Problem statement

The problem the paper wishes to address is as follows, given a set of correlated time series $X = \langle X^{(1)}, X^{(2)}, ..., X^{(i)}, ..., X^{(n)} \rangle$ out of which the first one is the dependent, target, series and the others are the independent series, using the past values of each series $\langle x_1^{(i)}, x_2^{(i)}, ...x_m^{(i)} \rangle$ where $1 \leq i \leq n$ and the relationship between the time series, we aim to predict the next p future values for the dependent series $\langle x_{a+l+1}^{(1)}, x_{a+l+2}^{(1)}, ..., x_{a+l+p}^{(1)} \rangle$, i.e. $\langle z_1, z_2, ..., z_p \rangle$. Based on state of art machine learning techniques two models are proposed that aim to reduce the prediction error of current baseline models.

The novelty of the paper consists in a new modular neural network model that takes advantage of both convolutional and recurrent properties for correlated time series prediction. By combining those two, the model outperforms the separated established models, and also other non-deep learning methods. Furthermore one of the models incorporates a multi-task learning method explained together with the loss function.

We also intent for the models to be robust, i.e. when given uncorrelated time series the models should be able to disregard the unnecessary data and perform similarly as when given just the target time series alone.

III. Related Work

Time series analysis and more precisely time series forecasting drew the attention from researchers in the recent years. Methods such as exponentially weighted moving average [8] or Autoregressive Integrated Moving Average (ARIMA) [9], used for forecasting univariate time series were continuously researched and improved since 1960. Even though the models are fairly old and simple by nature, they proved to be efficient for modeling linear time series and nowadays are commonly used as base line comparison for new models. One drawback that such models have is that they fail to take advantage of the interactions between time series, when more such interactions are available. For example, given a set of time series containing measurements of chemical concentrations from an industrial process, the chemical interaction can provide information that is discarded by these methods.

An alternative to such time series forecasting methods are neural networks. Using neural networks in this scope is not something new, in [10] we have seen such an idea being compared with Box-Jenkins, showing a promising way forward.

In [11] [12] the authors proposed a hybrid model that combines ARIMA with RNN. The model starts with ARIMA which is able to identify the linear dependencies from the data set. Afterwards the resulting residuals are fed to an RNN which should be able to capture the nonlinear dynamics. Tested on different data sets, the novel idea proved to be more efficient than both models alone.

Developed by *Hochreiter et al.* [13] Long short term memory (LSTM) is a new type of recurrent network. LSTM is capable of controlling what information gets learned or forgotten through structures named gates. It has been proven that such a model is able to effectively solve the vanish gradient problem for a more efficient learning [14].

In [15] the authors propose a novel model which comes as an extension to the LSTM. Their model starts by automatically extracting features from the data set using an LSTM auto-encoder which are then merged with the input vector and fed to an LSTM for predictions.

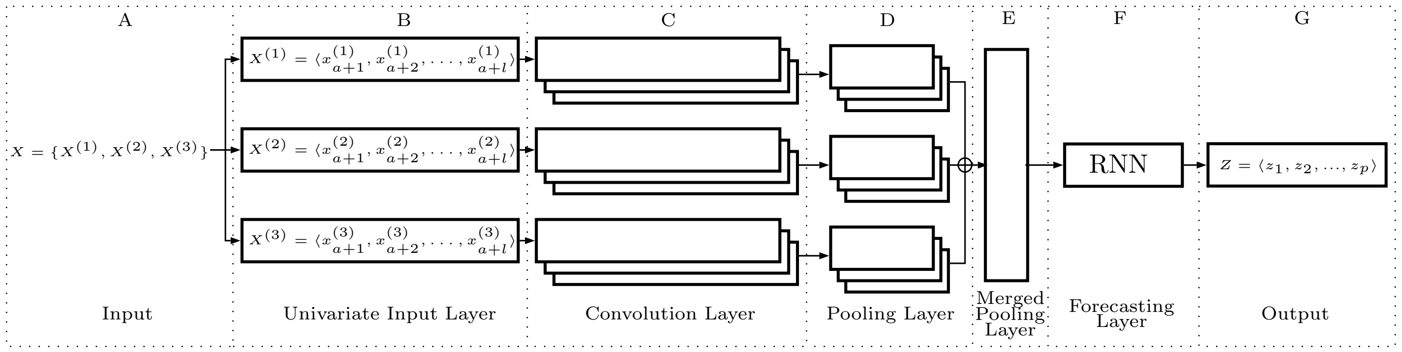
A
B
C
D
E
F
G
$X = \{X^{(1)}, X^{(2)}, X^{(3)}\}$
$X^{(1)} = \langle x^{(1)}_{a+1}, x^{(1)}_{a+2}, \dots, x^{(1)}_{a+l} \rangle$
$X^{(2)} = \langle x^{(2)}_{a+1}, x^{(2)}_{a+2}, \dots, x^{(2)}_{a+l} \rangle$
$X^{(3)} = \langle x^{(3)}_{a+1}, x^{(3)}_{a+2}, \dots, x^{(3)}_{a+l} \rangle$
RNN
$Z = \langle z_1, z_2, \dots, z_p \rangle$
Input
Univariate Input Layer
Convolution Layer
Pooling Layer
Merged Pooling Layer
Forecasting Layer
Output

Fig. 1: Modular Convolutional Recurrent Neural Network (MCRNN)

From their experiments adding the auto-encoder on top of the LSTM increased the prediction accuracy significantly.

Assad et al. present in [16] a boosting algorithm for single time series prediction using recurrent neural networks. They train multiple models iteratively, with each consecutive iteration putting more weight on fitting the data samples that were improperly fitted in the previous iteration. When the boosting reaches below a certain threshold the end result is calculated by averaging the results on all the models.

Similarly, [17] uses CNN in combination with LSTM, in what they call ConvLSTM, to predict one step ahead the amount of precipitation. The convolutions take place in between the recurrences of the LSTM combining the context of the previous recurrence with the new input. The results show an increased rate of correct predictions, lower false positives and generally better MSE scores. ConvLSTM takes as input a 2D image, not time series, but it was taken into consideration due to its similar idea of using a modular network network model.

In [18], *Pang et al.* propose a multivariate time series convolutional neural network (MTCNN) that uses a CNN to extract features from a multidimensional input before passing the results to a fully connected neural network layer. This allows them to generate better predictions of power consumption, in an industrial plant, compared to a regular CNN. The idea is similar to the model we prose since it uses a CNN feature extractor. Instead of connecting the features to a fully connected layer, we added a pooling layer first and fed the output to an RNN. In addition the learning procedure of the CNN filters in our case is affected by the reconstruction error which is not present in [18].

	Single Time Series	Multiple Time Series
Linear	[8], [9]	[19], [20]
Non linear	[11], [12], [15], [16], [17]	[6], [13], [18] **MCRNN, MAECNNRNN**

TABLE II: Related work comparison

IV. Proposed Model

For the basic structure of our proposed model we used a modular neural network. Within a regular feed forward neural network all the nodes inside a layer, including input layer, are connected to all the nodes in the following layers. In contrast, a modular neural network does not obey this restriction. This allows the creation of independent neural network, modules, each using their own segment of original input. The output of these modules can be collected by a fully connected network, called intermediary, that produces the final output of the network [21].

We have seen CNN being used with success for classifying images [22], videos [23] and audio signals [24] further more such networks have proven to be efficient in detecting shapes. The problem that we are tying to solve is a regression problem, which might require a different approach. The intuition is that CNN can extract significant features from the time series. Using those features further as input to an RNN, that can be used for regression, should provide a higher accuracy, due to reduced noise in the data.

By varying the type of both independent and intermediary networks and the training methods we proposed two models.

A. Modular Convolutional Recurrent Neural Network (MCRNN)

The first proposed model combines the CNN capabilities of recognizing patterns and shapes from data with RNN which is able of capturing short term dependencies and forecasting future values. The graphical representation of the model can be observed in Fig. 1.

The model takes as input a multidimensional array that represents multiple time series, where each time series has a total of l measurements. For each time series in the original input $X^{(1)}, X^{(2)}$, and $X^{(3)}$ we create an independent CNN that is going to receive as input a $1 \times l$ matrix as described in column B.

We then apply convolutions on each time series individually (column C). In the convolution layer, when applying the filters we have the option to choose how many filters we want and their size. We apply γ, e.g., 3 in Fig. 1, filters. Each filter is going to yield a $1 \times l$ matrix, so in total we will have one $\gamma \times 1 \times l$ cube for each time series. After the convolution is done, we apply the rectifier function (ReLU) to ensure that all the resulting values are positive. Note that during some preliminary tests we observed that removing the activation function after the convolution layer improved the results.

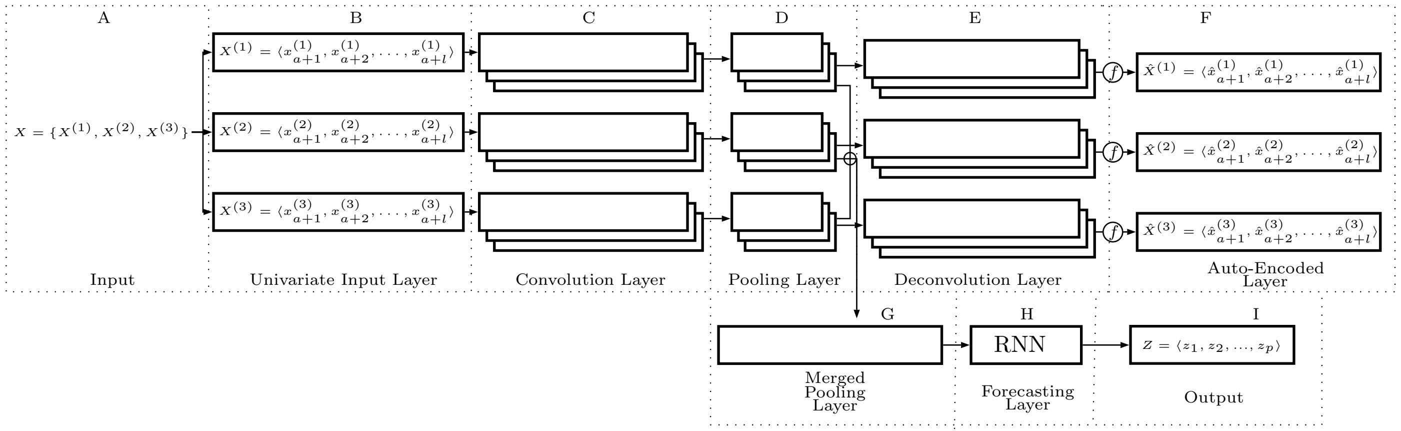
A
B
C
D
E
F
$X = \{X^{(1)}, X^{(2)}, X^{(3)}\}$
$X^{(1)} = \langle x^{(1)}_{a+1}, x^{(1)}_{a+2}, \ldots, x^{(1)}_{a+l} \rangle$
$X^{(2)} = \langle x^{(2)}_{a+1}, x^{(2)}_{a+2}, \ldots, x^{(2)}_{a+l} \rangle$
$X^{(3)} = \langle x^{(3)}_{a+1}, x^{(3)}_{a+2}, \ldots, x^{(3)}_{a+l} \rangle$
f
$\hat{X}^{(1)} = \langle \hat{x}^{(1)}_{a+1}, \hat{x}^{(1)}_{a+2}, \ldots, \hat{x}^{(1)}_{a+l} \rangle$
$\hat{X}^{(2)} = \langle \hat{x}^{(2)}_{a+1}, \hat{x}^{(2)}_{a+2}, \ldots, \hat{x}^{(2)}_{a+l} \rangle$
$\hat{X}^{(3)} = \langle \hat{x}^{(3)}_{a+1}, \hat{x}^{(3)}_{a+2}, \ldots, \hat{x}^{(3)}_{a+l} \rangle$
Input
Univariate Input Layer
Convolution Layer
Pooling Layer
Deconvolution Layer
Auto-Encoded Layer
G
H
I
RNN
$Z = \langle z_1, z_2, \ldots, z_p \rangle$
Merged Pooling Layer
Forecasting Layer
Output

Fig. 2: Modular Auto Encoder Convolutional Neural Network + Recurrent Neural Network (MAECNNRNN)

Afterwards, in column D, max or average pool is performed on each matrix in order to capture the most representative features. The pooling operator reduces the size of each cube by half $\gamma \times 1 \times \frac{l}{2}$ since we used a 1×2 window with a stride of 2. As the name implies when applying max pooling, the biggest value from a certain window, 2 in our case, is selected, while the latter would perform an average.

After the pooling layer we have a total of γ matrices of size $1 \times \frac{l}{2}$. All cubes are going to be merged into a 1 dimensional vector of size $n = |X| \times \gamma \times 1 \times \frac{l}{2}$. The way we merged the matrices is as follows: starting with the first cube we take the first value from the first dimension, then from the second dimension etc. The same process is applied afterwards for each cube.

Next we feed the merged pooling layer to an RNN which is responsible for forecasting the next p step ahead for the target time series, $X^{(1)}$ in our case. An example of possible input for the forecasting layer is presented in Fig. 6. Note that the forecasting layer F can be replaced with any other neural network model. For the experiments we used a vanilla sigmoid-RNN implementation.

MCRNN can be adapted to work with any n number of time series by creating a separate network branch for each time series in columns B, C, D. In our experiments we tested MCRNN with one, two, respectively three time series. In addition the models can be easily modified to predict all time-series if required.

Cost function and Optimization: The cost function for MCRNN is described by equation (1) where $Error(\cdot, \cdot)$ is in our case the mean square error function (MSE) that reflects the model accuracy.

In equation (1), the loss function J_1 takes as input: the set of weights matrices W' of the Forecasting layer in column F (for MCRNN Fig.1 and column H for MAECNNRNN Fig.2) and Convolution Layer in column C; the set of biases b' from the same columns; the target time series $X^{(1)}$; and the predicted time series Z in order to compute the forecasting error. On the right side of the equation we have the average of l *Error*s where each error represents the discrepancy between the predicted value z_i and the ground truth $x^{(1)}_{a+l+i}$ at time tick i.

$$J_1(W', b', X^{(1)}, Z) = \frac{1}{p} \sum_{i=1}^{p} Error(z_i, x^{(1)}_{a+l+i}) \qquad (1)$$

Eq.: Prediction loss function [18]

Learning is performed using backpropagation by minimizing the error term of the prediction. For the experiments the Adam optimizer was selected. In [25] Adam optimizer has been proven to be more efficient for gradient-based optimization problems than classical approaches such as Stochastic gradient descent (SGD).

When learning the weights matrices W, a fraction η multiplied with the partial derivative of the cost function with respect to the weights is subtracted from the old W. η denotes the learning rate, a hyper-parameter that was manually tuned for each data set individually for optimal results.

$$W = W - \eta \frac{\partial J(W, b, X, Z)}{\partial W} \qquad (2)$$

Eq.: Weights update function [18]

B. Modular Auto Encoder Convolutional Neural Network + Recurrent Neural Network (MAECNNRNN)

The second proposed model Fig. 2, similar in spirit with the first one, also combines CNN with RNN. The main difference between the two models is that we replaced the Convolutional and the Pooling layer (columns C and D in Fig. 1) with an CNN auto-encoder (columns C, D, E, F in Fig. 2). As the name implies the auto-encoder is responsible for learning the data representation in an unsupervised manner. The intuition behind this model is that the auto-encoder will learn more robust features, and how to ignore the outliers. In addition the auto-encoders work as a regularization term that will prevent the model from over-fitting.

In order for the auto-encoder to reconstruct the original time series a Deconvolution Layer was added (column E). The Deconvolution Layer is going to take the resulting cubes from the Pooling Layer as input and deconvolve each cube into γ matrices that have the same size as the ones from column C. Afterwards by applying the $sigmoid$ function on the matrices the output will correspond to the target output, the reconstructed data for each time series $\hat{X}^{(1)}$, $\hat{X}^{(2)}$, and $\hat{X}^{(3)}$, where $\hat{X}^{(i)} = \langle \hat{x}^{(i)}_{a+1}, \hat{x}^{(i)}_{a+2}, \ldots, \hat{x}^{(i)}_{a+l} \rangle$.

Another important difference between the two models is the learning problem. For the first model we tried to minimize one error, the one produced by the forecasting layer (column F Fig. 1). For the second model we are minimizing a multi objective function. The loss function consists of a sum of $n+1$ errors. During backpropagation the error of each deconvolution layer (column E) is only affected by the reconstruction error, while the filters (column C) are affected both by the reconstruction error and the prediction error.

Cost function: Equation (3) represents the model's capabilities to reconstruct the original data. It is a sum over all the time series reconstruction errors, where the reconstructed value is defined by $\hat{x}^{(k)}_{a+i}$ and the ground truth by $x^{(k)}_{a+i}$ for each time stamp $(a+i)$.

In equation (3), the loss function J_2 takes as input: the set of weights matrices W'' of the Deconvolution Layer in column E and Convolution Layer in column C; the set of biases b'' from the same columns; all time series X; and the reconstructed time series $\hat{X}$ in order to compute the reconstruction error. On the right side of the equation we have the average sum for each p $Error$s where each error represents the discrepancy between the reconstructed value $\hat{x}^{(k)}_{a+i}$ and the observed value $x^{(k)}_{a+i}$ at time tick i for each time series k.

$$J_2(W'', b'', X, \hat{X}) = \frac{1}{l} \cdot \frac{1}{n} \cdot \sum_{k=1}^{n} \sum_{i=1}^{l} Error(\hat{x}^{(k)}_{a+i}, x^{(k)}_{a+i}) \quad (3)$$

Eq.: Reconstruction error function

The loss function for MAECNNRNN is $J = J_1 + J_2$. Since the loss function is represented by a bi-objective function this makes MAECNNRNN a multi task learning model, where the model tries to minimize the final predictions made by the RNN and CNN capabilities of reconstructing the data.

Note that MAECNNRNN, similar to MCRNN, can be adapted to work and predict any n number of time series.

V. Testing scenarios

For the experiments we have selected two data sets, sewage chemicals concentration and Google trends.

The first data set was provided by a sewage treatment plant from Aalborg, Denmark, and it represents a collection of time series. Each time series corresponds to a chemical concentration measurement taken at an interval of two minutes from each of the six aeration tanks within the facility. The chemicals monitored are NH_4, NO_3, and O_2. Furthermore, from each of the 6 tanks we take three time periods resulting in a total of eighteen smaller data sets for the experiments.

The second data set was taken from Google Trends and it represents the search popularity of related terms on Google. The set contains two time series with the dependent (target) time series being the term *potatoes* and independent time series being the term *brown sugar*. The Pearson correlation coefficient of the two terms is 0.9601.

As mentioned previously, we have computed the pairwise Spearman correlation coefficients between the dependent time series and all the remaining independent time series and the correlations discovered were significant enough for our purposes. The chosen dependent variables are NH_4 for the chemical concentrations and *potatoes* of the second data set.

When learning the models we segmented our data sets into multiple test cases using a sliding window. Fig. 3 illustrates the process. The input data contains l time measurements while target data contains p values ahead denoted in table IV. We decided to segment the data set into multiple test cases so we can learn by batches in order to boost the learning speed while still maintaining good accuracy. In addition we normalized the data by scaling individual time series between 0 and 1. Evaluation of the model was used on a reserved 16% of the window, referred to as testing data.

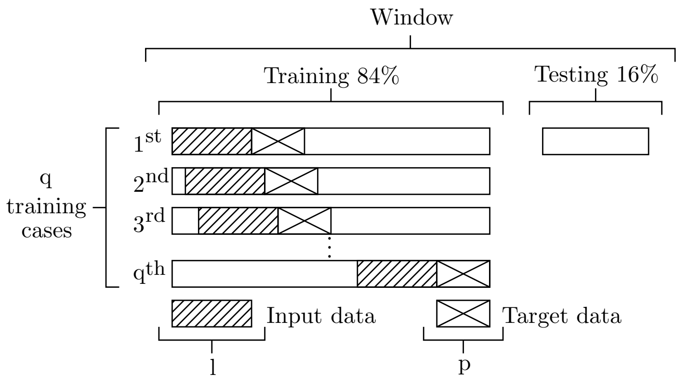

Fig. 3: Training vs Testing

Validation and hyper-parameters selection on the sewage data set for all algorithms was performed on tank 4 section 1. This data set was excluded from all the results presented later in the paper. The reason for exclusion is to eliminate any conflict generated by the intersection of validation data with testing data.

There are several parameters that can vary when testing the models: n the number of time series in X, $n = |X|$; the length of every time series in the input l; number of ticks to forecast in the immediate future p. Table IV shows the values of those parameters. The complete list of configurations that are executed is presented in table III

Within the convolution layers of our models, we also varied the number and the size of the filters per convolution. Also adjusted was the number of convolution layers for each time series. This is shown in table V.

Notation	n	l	p
2_10_100	$n = 2$	$l = 10$	$p = 100$
2_50_100	$n = 2$	$l = 50$	$p = 100$
2_100_100	$n = 2$	$l = 100$	$p = 100$
2_50_1	$n = 2$	$l = 50$	$p = 1$
2_50_25	$n = 2$	$l = 50$	$p = 25$
2_50_50	$n = 2$	$l = 50$	$p = 50$

TABLE III: Experimental configurations for sewage data set

Parameter name	Values
Number of time series	$n = \{1, 2, 3\}$
Input length per time series	$l = \{10, 50, 100\}$
Prediction steps	$p = \{1, 25, 50, 100\}$

TABLE IV: Problem parameters

Parameter name	Values
Convolution Layers	$c = \{1, 2, 3\}$
Filters per convolution	$f = \{2, 3, 4, 5, 8, 10, 16\}$
Filter size	$\{1, 2, 3, 5, 10\}$
RNN hidden size	$\{3, 4, 5, 6\}$

TABLE V: Solution parameters

Table VI shows algorithms tested together with the base line comparison algorithms. The intuition is that models that use correlation between time series are capable of generating better results, as they are provided more relevant information. Each algorithm is tested using two correlated time series to show individual capabilities of modeling correlations.

In order to test the robustness of our models when given a set of uncorrelated time series, a random time series was generated and used together with a regular time series. The intuition behind the comparison is that correlated series yield a better prediction than non-correlated ones.

Our algorithms are compared with other established or state of art algorithms.

Yesterday is a linear method of regression that propagates the last know value of the time series for the entire prediction window. It is commonly used in financial time series prediction [26]. ARIMA is also a commonly used regression model for time series applied in various fields of industry [9].

Explained in chapter III, MTCNN is multivariate and nonlinear time series prediction model. It was selected as state of art comparison for its related scope with our problem statement. We were unable to find an implementation of MTCNN by the original authors or otherwise, so we implemented it ourselves in Python based on the model description chapter of the paper [18]. Adjustments were made on the number of fully connected hidden layers, number of nodes in those layers, and learning rate for optimal results.

RNN and LSTM were selected as baseline comparisons for their established deep learning regression uses.

Algorithm	Multiple Time Series	Nonlinear
Yesterday	no	no
ARIMA	no	no
MTCNN	yes	yes
RNN	yes	yes
LSTM	yes	yes
MCRNN	yes	yes
MAECNNRNN	yes	yes

TABLE VI: Classification of algorithms

VI. Model evaluation

For evaluating the models we have selected two error measurements, root mean square error (RMSE) and mean absolute percentage error (MAPE).

RMSE is a popular error measurement used for assessing a model's accuracy when working with Neural Networks in the scope of regression. The formula is based on the euclidean distance between the predicted z_t and the observed value $x^{(1)}_{a+l+t}$ at every time tick t. Despite its popularity this error measurement has two major drawbacks. RMSE tends to heavily penalize large errors and is hard to interpret [27].

$$RMSE = \sqrt{\frac{1}{p}\sum_{t=1}^{p}(z_t - x^{(1)}_{a+l+t})^2} \tag{4}$$

The second error measurement, MAPE, was chosen because of its absolute nature independent of the scale of measurements. This makes it easier to interpret for people not familiar with the data. One big drawback of this method is that it does not work when the data sets contain measurements equal to 0, since this results in a division by 0. This is not an impediment for us since the predicted time series, the dependent one, does not contain such measurements.

$$MAPE = \frac{100}{p}\left|\sum_{t=1}^{p}\frac{x^{(1)}_{a+l+t} - z_t}{x^{(1)}_{a+l+t}}\right| \tag{5}$$

Evaluation of the model was performed in five stages as follows: tuning of the solution parameters; comparison between linear and non linear; comparison between using one or multiple time series; comparison based on various of input and output sizes; overall comparison.

A. Tuning of the solution parameters

We begin by adjusting hyper-parameters for each algorithm for optimal results. These optimizations are based on the evolution of the MSE after each epoch in order to achieve models that are guarded against overfitting and underfitting. The epoch size was set to 2000 for both data sets. We repeated this process for each possible configuration illustrated in table III (6 times for the sewage data set).

Due to the different nature of the second data set and the the amount of time required to fine tune configurations the

Config/Alg	YESTERDAY	RNN	LSTM	MTCNN	MCRNN	MAECNNRNN	ARIMA
2_10_100	1.02± 0.36	0.69± 0.30	0.71± 0.28	0.60± 0.31	0.64± 0.33	**0.57± 0.30**	1.42± 1.00
2_50_100	1.00± 0.36	0.74± 0.28	0.79± 0.29	0.64± 0.35	0.74± 0.35	**0.58± 0.31**	1.15± 0.47
2_100_100	1.08± 0.59	0.83± 0.37	0.83± 0.38	0.73± 0.44	0.95± 0.50	**0.66± 0.38**	1.18± 0.66
2_50_1	0.09± 0.03	0.21± 0.13	0.15± 0.09	0.14± 0.08	0.13± 0.08	0.20± 0.13	**0.05± 0.02**
2_50_25	0.93± 0.22	0.44± 0.22	0.45± 0.24	0.45± 0.25	0.42± 0.26	**0.38± 0.21**	0.74± 0.25
2_50_50	1.03± 0.28	0.66± 0.30	0.55± 0.31	0.57± 0.34	0.54± 0.29	**0.47± 0.28**	1.03± 0.40

TABLE VII: RMSE experiment results for sewage data set. For MAPE results see table X

Config/Alg	YESTERDAY	RNN	LSTM	MTCNN	MCRNN	MAECNNRNN	ARIMA
2_50_25	1.02	0.66	0.64	0.60	**0.53**	**0.53**	1.04
2_50_50	1.10	0.67	**0.60**	0.73	**0.60**	0.62	1.18

TABLE VIII: RMSE experiment results for Google Trends data set. For MAPE results see table XI

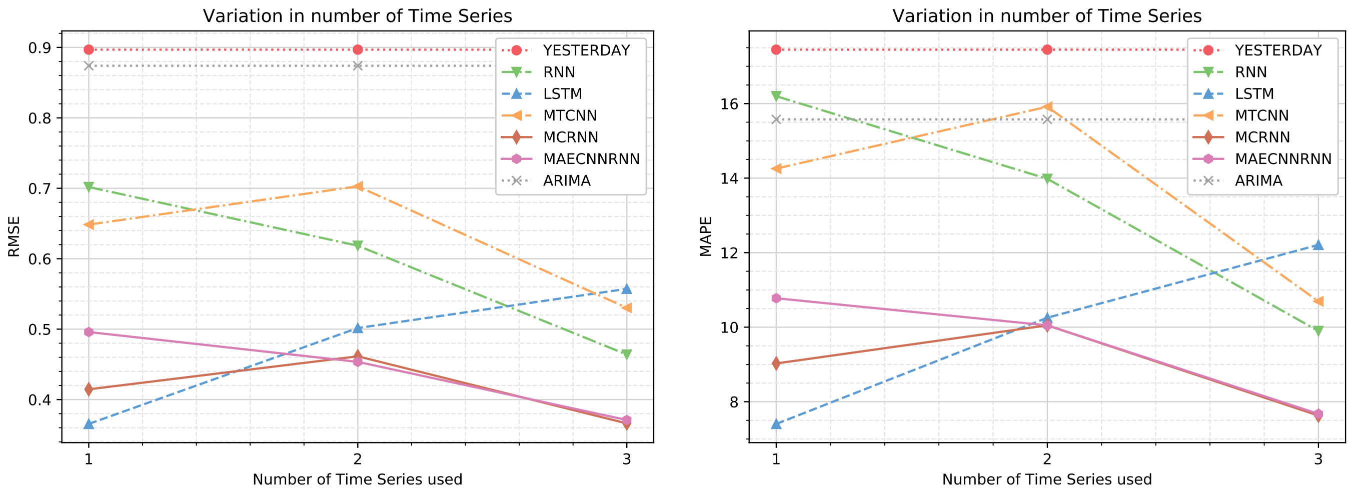

Fig. 4: RMSE (left) and MAPE (right) variation on sewage data set for configurations 1_50_25, 2_50_25, 3_50_25

Google Trends data set was ran on 2 configurations: 2_50_25 and 2_50_50. They were selected because they offer a balanced amount of input and output data while taking advantage of the correlated time series. Adjustments are made to the hyper-parameters for these configurations for each algorithm.

B. Linear and non linear

The first assumption tested is that non linear models are more suitable for regression in situations where the data might convey non linear relations between time series. All deep learning models used are inherently considered non linear.

Table VII and associated figures 8, 12, 13, 9 and 10 proves that all non linear models are more suitable for long term predictions. However, linear models such as ARIMA and Yesterday are more suitable for short term prediction Fig. 11.

C. Using one or multiple time series

The second assumption tested is that using multiple series would generate an inherit advantage in regression due to more information that can be extracted from the relations between the time series.

We check this assumption by varying the problem parameter n. The other two problem parameters are kept constant with $l = 50$ and $p = 25$ while n was varied between one, two and three time series. Due to limited time the configurations are ran on only on the first sewage data set.

Fig. 4 show the results of running these experiments. It can be observed that increasing the number of time series resulted in better RMSE and MAPE scores for MAECNNRNN. Both MCRNN and its competitor MTCNN, show a slight increase in error scores for two time series, but when the third time series was added the performance increased significantly.

For our two models MCRNN and MAECNNRNN we also tested their behavior when given uncorrelated time series, with the assumption that they are able to filter out unnecessary data. We do this by feeding the algorithm a set of regular target time series and a random time series. The random time series was prior created and tested so that there would not be a significant correlation between it and the target series.

Table IX confirms our assumptions. It can be observed that both proposed models are performing better when exploiting the correlation between series. Note that MAECNNRNN is

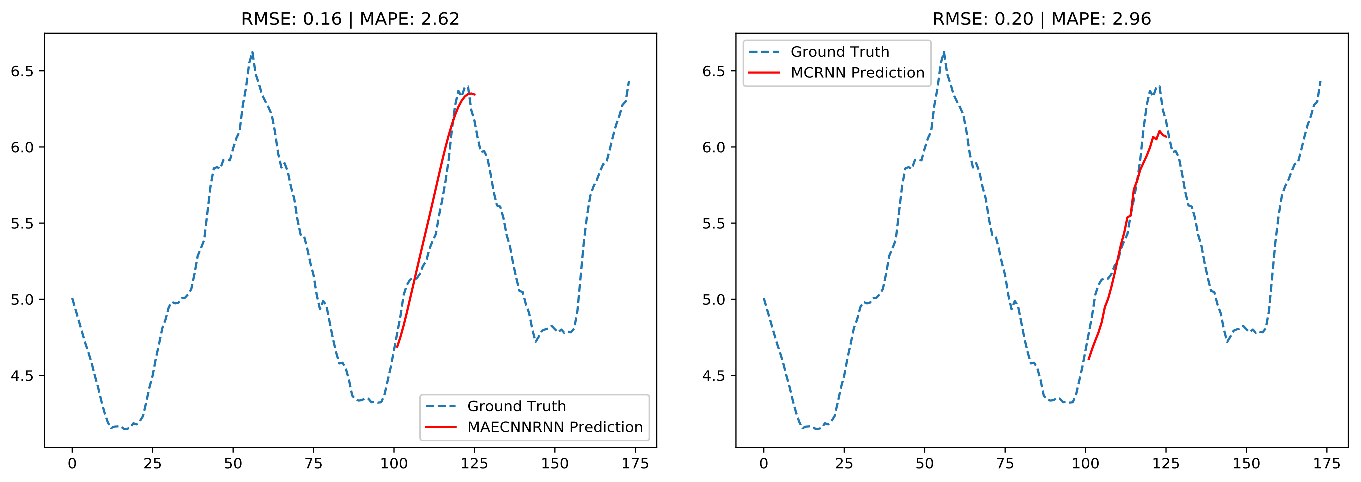

Fig. 5: Prediction example for MAECNNRNN (left) and MCRCNN (right) for configuration 2_50_25 on aeration tank 4 from sewage data set

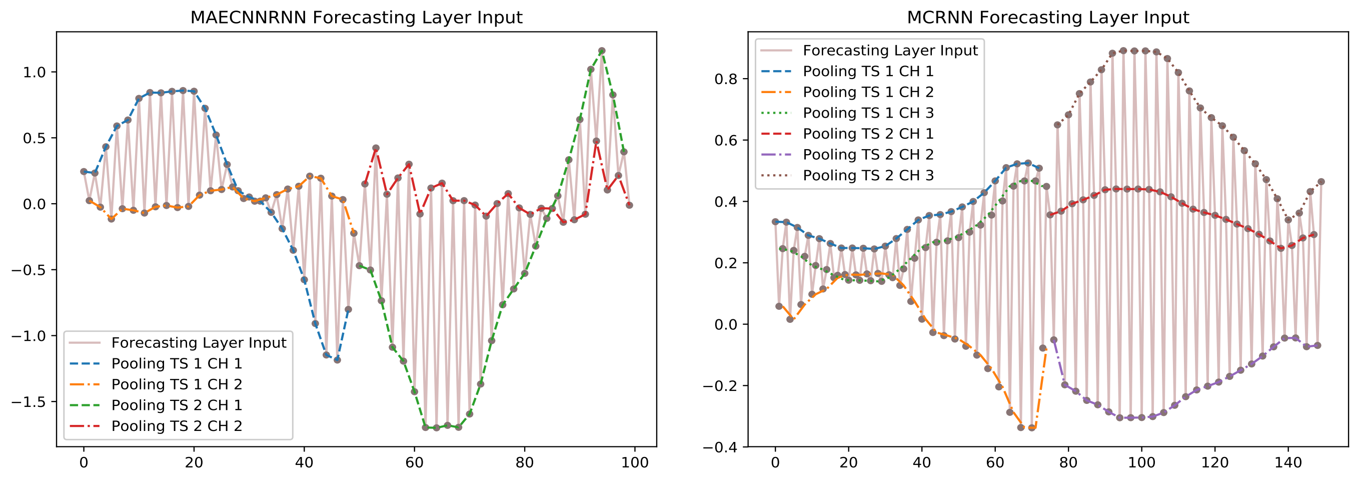

Fig. 6: Input for Forecast Layer MAECNNRNN (left) and MCRNN (right) for configuration 2_50_25 on aeration tank 4 from sewage data set

	MCRNN	MAECNNRNN
Single target TS	10.8	**10.7**
With a correlated TS	**8.3**	9.1
With an uncorrelated TS	13.0	**10.7**

TABLE IX: Dealing with uncorrelated time series, MAPE

more robust. Due to its auto-encoders the model is able to filter out the redundant information, random time series, from the input and maintain the same error as the single target time series configuration. The parameters used for these tests are $n = 1$ or $n = 2$, $l = 200$, and $p = 100$ the tests are also ran exclusively on the first sewage data set.

D. Comparison based on various of input and output sizes

Table III shows six configurations for the remaining two problem parameters l, p.

Varying l, the length of the input time series, we discovered that our model MAECNNRNN IV-B outperforms all its competitors. The results are shown in Fig. 14. The other two parameters n, p were set to 2 and 100 respectively.

The last problem parameter we tested is p, the length of the model output. Fig. 15 shows that both of our algorithms perform best on longer predictions periods. However it can also be observed that ARIMA and Yesterday perform best when $p = 1$, which is expected as chemical concentrations readings can not change drastically within a time span of 2 minutes.

E. Overall comparison

Fig. 8, 9, 10, 11, 12, and 13 present the overall comparisons between the RMSE and MAPE scores of all algorithms on the sewage data sets. The values are averaged on all of the 17 data sets and shown together with the standard deviation.

Our algorithms perform well on almost all the configura-

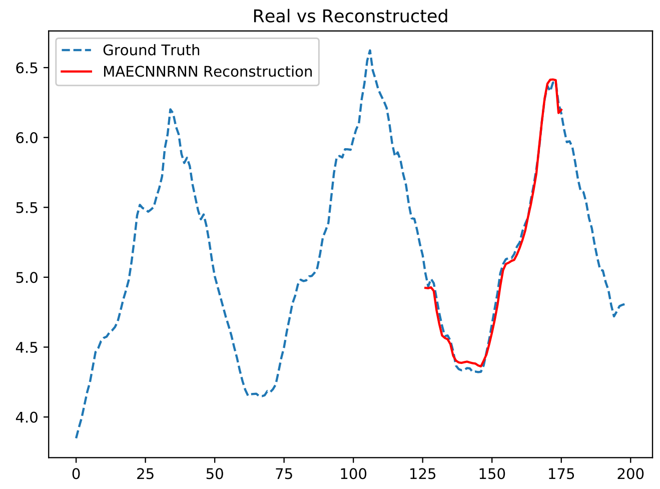

Fig. 7: Reconstruction example for MAECNNRNN for configuration 2_50_25 on aeration tank 4 from sewage data set

tions, with the closest competitor MTCNN also offering good results. In contrast to MTCNN, both of our algorithms present as the intermediary network a vanilla RNN that uses information propagated from previous predictions for the current one. We believe this offers our models the edge when it comes forecasting. A second major difference between our proposed models and MTCNN is the complexity, MTCNN requiring multiple fully connected layers resulting in a much greater state space.

Fig. 5 is an example of predictions made by our two models when $n = 2$, $l = 50$, and $p = 25$. It can be observed that MAECNNRNN does a better job of predicting the amplitude of the spike, but is smoother, meaning it ignores the small distortions at the top. We consider this to be a consequence of the auto-encoding process that might value the correct amplitude more and ignores the small distortions. This behavior can be observed in Fig. 7 where in red is reconstructed input generated by the auto-encoder. In contrast MCRNN, predicts a lower amplitude but does capture the distortions at the top.

Fig. 6 shows the input of the Forecast Layer for our two models. For MAECNNRNN we used 2 filters, $\gamma = 2$, and for MCRNN 3 , $\gamma = 3$. As mentioned previously this hyper parameter was chosen for optimal results. In the figure, 'Pooling TSi CHj' stands for the output of the Pooling Layer of the time series $X^{(i)}$ from channel j. For MAECNNRNN each filter yields a more human readable result, with one of the filters focused on a general representation of the input while the other tries to model the residuals. Without the auto-encoding layer, MCRNN is granted more liberty in selecting whatever features it finds more usable, resulting in less human readable input.

The two vanilla recurrent neural networks, RNN and LSTM perform similarly. While LSTM is expected to outperform RNN, our results might have been influenced by the manual hyper-parameter adjustments.

As previously mentioned, Yesterday and ARIMA perform the best when $p = 1$. Compared with each other, the simplicity of Yesterday does not allow it generated massive errors resulting better RMSE and MAPE scores compared with ARIMA. This might be due to the fact that ARIMA requires a lot of parameter adjustment.

Table VIII and XI present the results of all the algorithms on the Google Trends data set. MCRNN performs the best while MAECNNRNN comes second on configuration 2_50_25. As observed in Fig. 5 and Fig. 7 MAECNNRNN tends to overgeneralize. Because of Google irregularities, it slightly decreases the model accuracy but still remains relatively accurate.

VII. Conclusion

Both neural networks and time series prediction are increasingly popular fields of research and development. In this paper we presented MCRNN IV-A and MAECNNRNN IV-B, two neural network based models for the purpose of *correlated time series prediction*. The basic intuition for the models is to utilize CNN capabilities of capturing features and feed them to an RNN which is capable of identifying short term dependencies in the data.

Empirically tested on a large real world data set containing chemical measurements from a treatment plant both proposed models showed increased accuracy compared with state of art algorithms. Both models proved themselves to be robust and confirm all our assumptions such that correlation improves accuracy.

In conclusion, the paper achieved its goal of accurate correlated time series prediction offering novel models that combine convolutional and recurrent neural networks.

VIII. Future work

Due to the limited period of time we only conducted experiments on sewage and Google trends data sets. Even though our models performed overall better than the competition on the sewage data sets we can observe that for some specific configurations that was not the case. Extensive experiments on multiple data sets are required to identify the cause of the problem.

One drawback of the model is the amount of time required to fine tune it. There are a lot of hyper-parameters that need to be properly selected so that the model will perform optimally. This process is time consuming since it requires the models to run a couple of times. An automatic way of selecting the parameters could be beneficial or improving the model's running time.

The basic structure of our models is a modular neural network. Both presented models, MCRNN and MAECNNRNN, use an RNN as an intermediary network for forecasting and various of CNN as independent networks. Changing the nature of the independent and intermediary networks might result in new models that offer improvements for certain data sets or predictions tasks.

The presented models were tested and evaluated in the scope of regression. By slightly modifying the model and replacing the MSE error function with cross-entropy function the new model is capable of doing classification. Further experiments are required to confirm or contradict the proposed models capabilities of classifying.

When applying the filters column C in figures Fig. 1 and Fig. 2 we have observed that removing the activation function (ReLU) improved the results on some data sets. More experiments are required to identify a pattern or an underlying process.

Acknowledgements

We would like to thank our supervisor Prof. Bin Yang for his guidance throughout the project period.

We also thank the treatment company for providing us the data set, together with explanations to the processes involved.

Config/Alg	YESTERDAY	RNN	LSTM	MTCNN	MCRNN	MAECNNRNN	ARIMA
2_10_100	25.51±12.99	19.17±12.01	19.88±13.52	14.46± 7.73	16.87±10.54	**14.22± 7.92**	26.57±15.11
2_50_100	25.95±13.64	22.93±17.10	24.09±17.91	15.65± 8.23	23.08±20.31	**15.28±10.09**	26.50±16.75
2_100_100	28.25±14.34	25.61±19.25	27.30±24.59	19.78±12.22	29.80±23.26	**18.32±11.82**	29.81±17.22
2_50_1	2.21± 1.09	4.76± 3.05	3.63± 2.19	3.34± 1.84	3.00± 1.63	5.04± 3.36	**0.98± 0.41**
2_50_25	23.61±10.77	11.15± 6.09	11.81± 8.59	11.44± 5.66	10.27± 5.64	**9.66± 5.57**	16.95±10.95
2_50_50	25.80±12.43	20.65±16.46	15.61±11.50	14.64± 8.28	15.52±12.51	**11.57± 5.99**	24.67±15.99

TABLE X: MAPE experiment results for sewage data set

Config/Alg	YESTERDAY	RNN	LSTM	MTCNN	MCRNN	MAECNNRNN	ARIMA
2_50_25	392.66	**337.09**	403.97	469.04	347.68	395.33	700.16
2_50_50	407.11	368.47	**363.02**	462.82	425.61	484.09	671.68

TABLE XI: MAPE experiment results for Google Trends data set

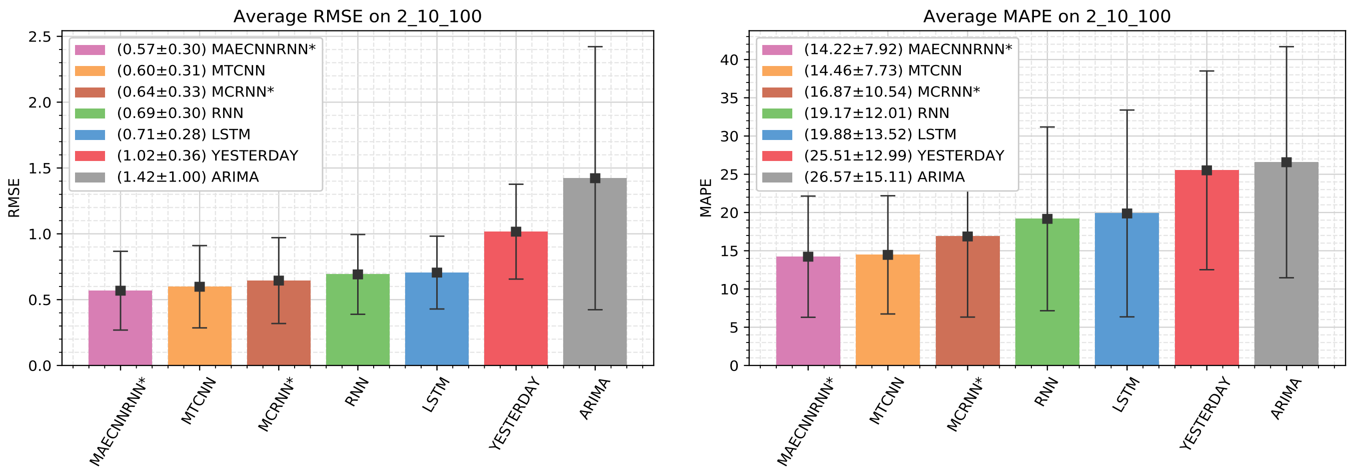

Fig. 8: Average RMSE (left) and MAPE (right) on sewage data set for configuration 2_10_100

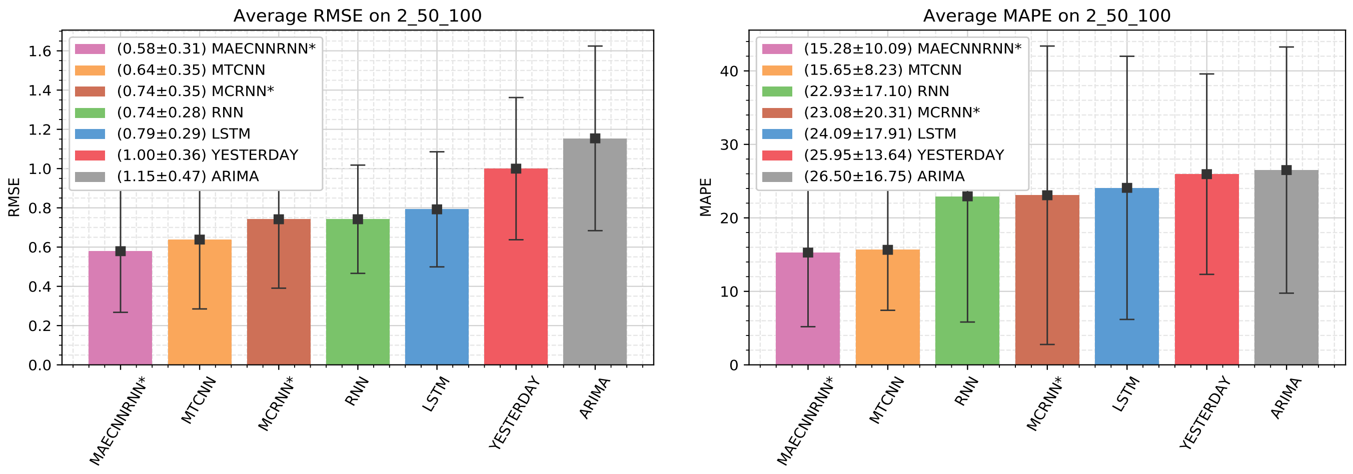

Fig. 9: Average RMSE (left) and MAPE (right) on sewage data set for configuration 2_50_100

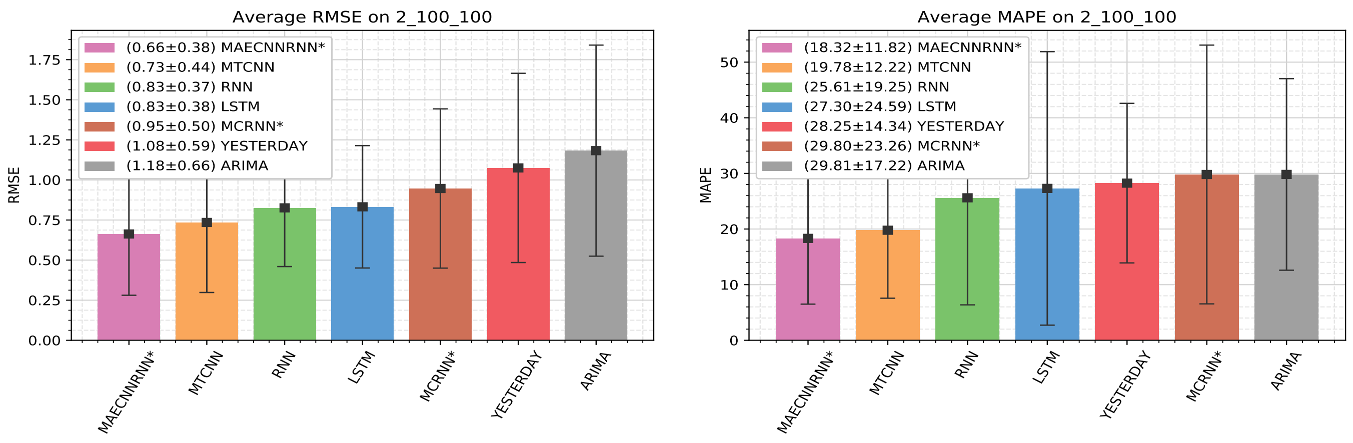

Fig. 10: Average RMSE (left) and MAPE (right) on sewage data set for configuration 2_100_100

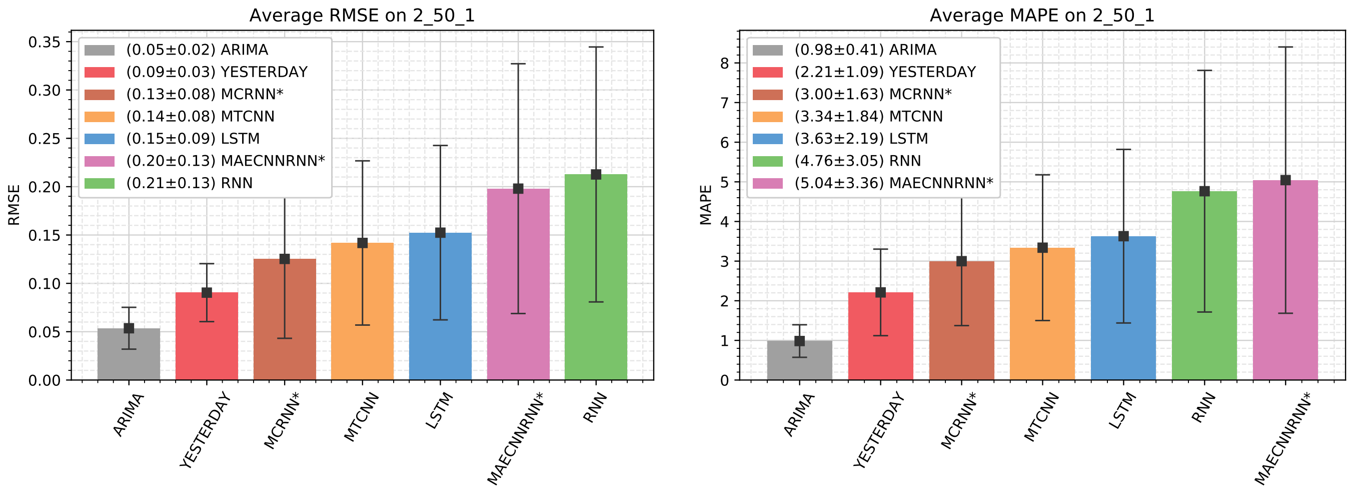

Fig. 11: Average RMSE (left) and MAPE (right) on sewage data set for configuration 2_50_1

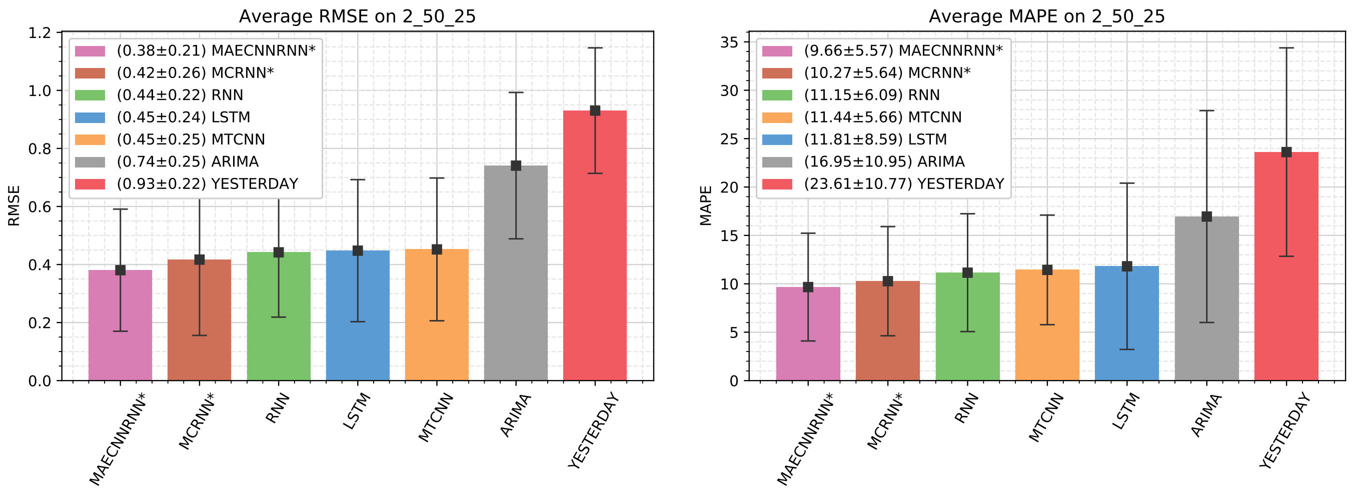

Fig. 12: Average RMSE (left) and MAPE (right) on sewage data set for configuration 2_50_25

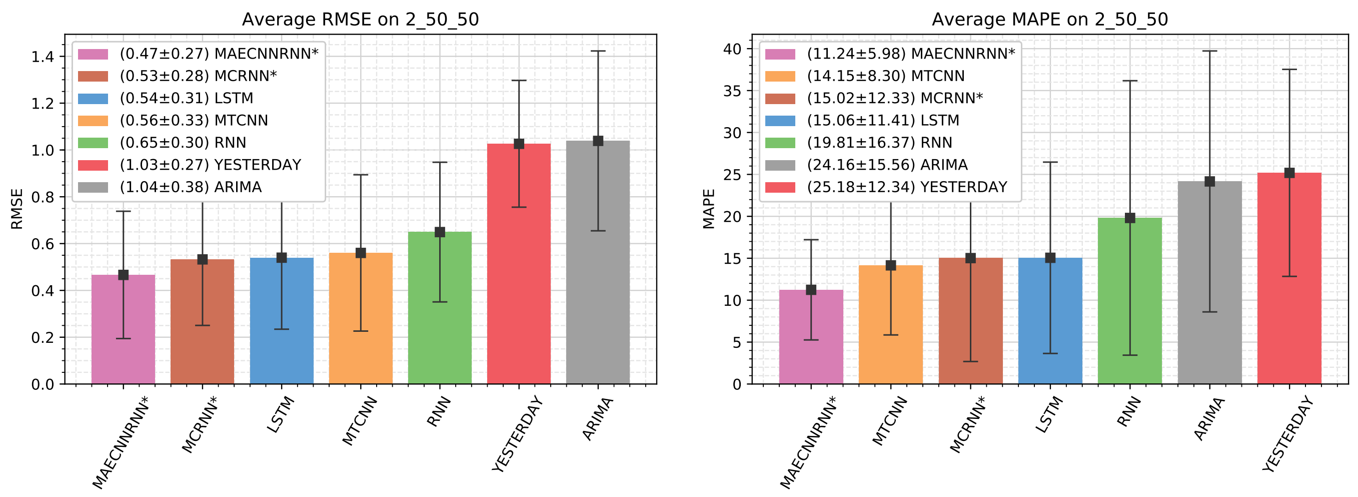

Fig. 13: Average RMSE (left) and MAPE (right) on sewage data set for configuration 2_50_50

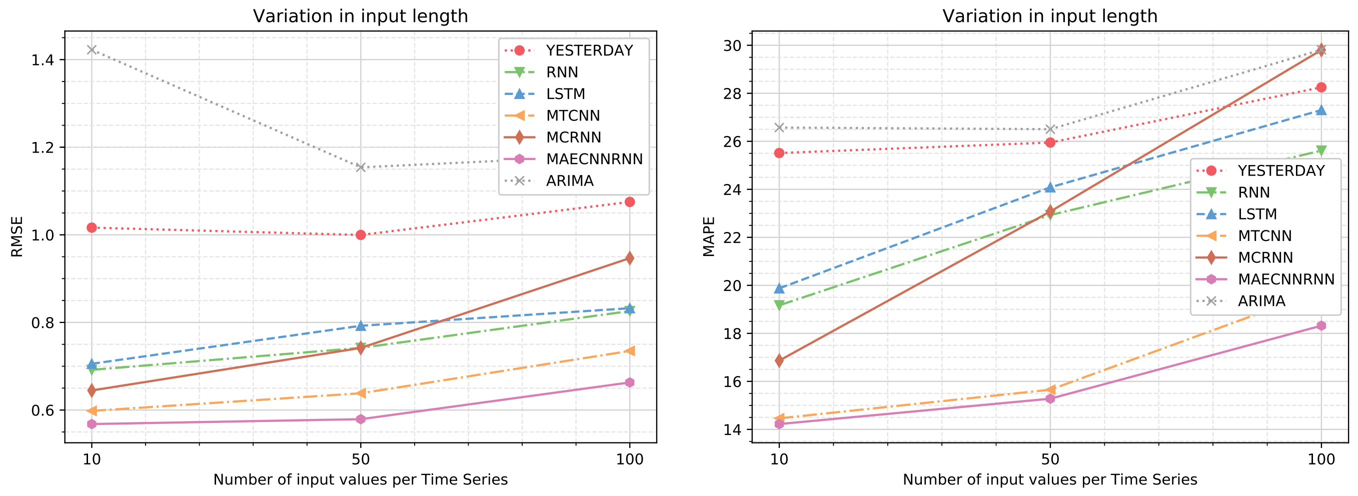

Fig. 14: Average RMSE (left) and MAPE (right) on sewage data with various input size

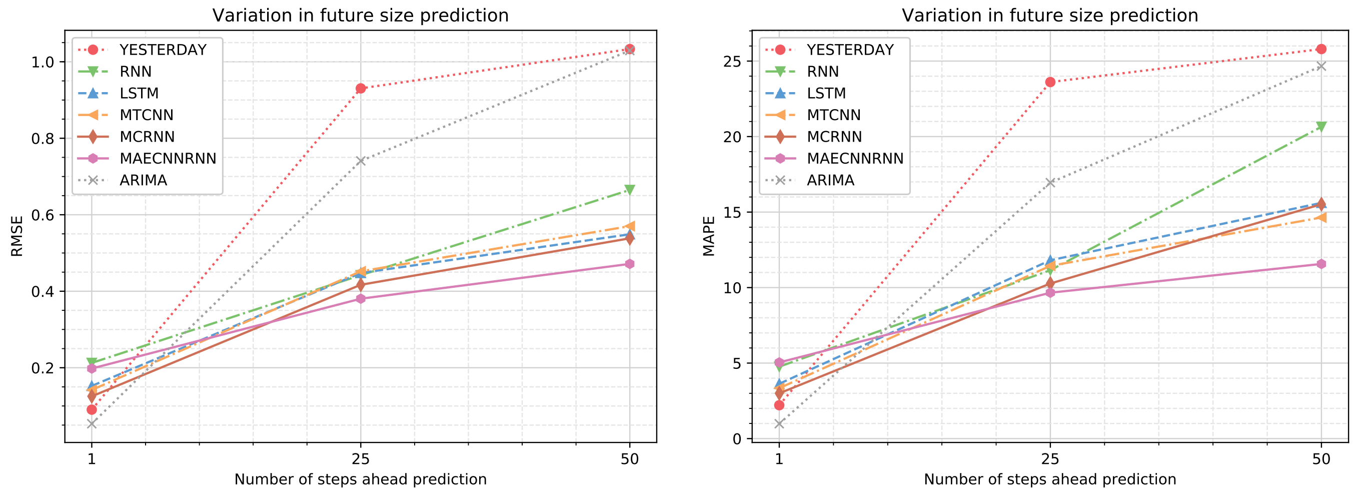

Fig. 15: Average RMSE (left) and MAPE (right) on sewage data with various prediction size

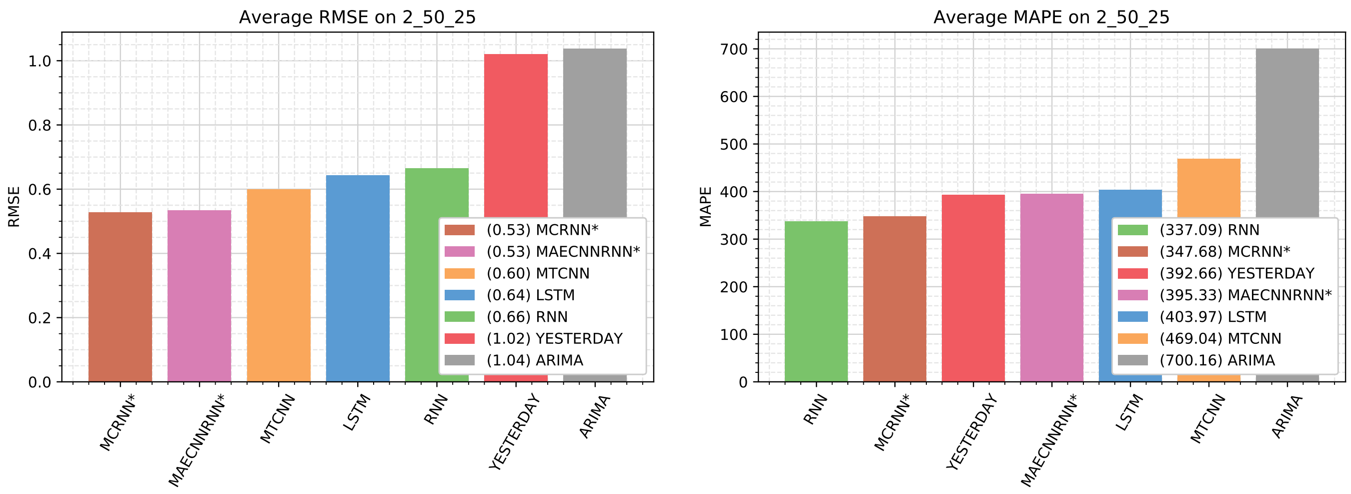

Fig. 16: Average RMSE (left) and MAPE (right) on Google Trends data set for configuration 2_50_50

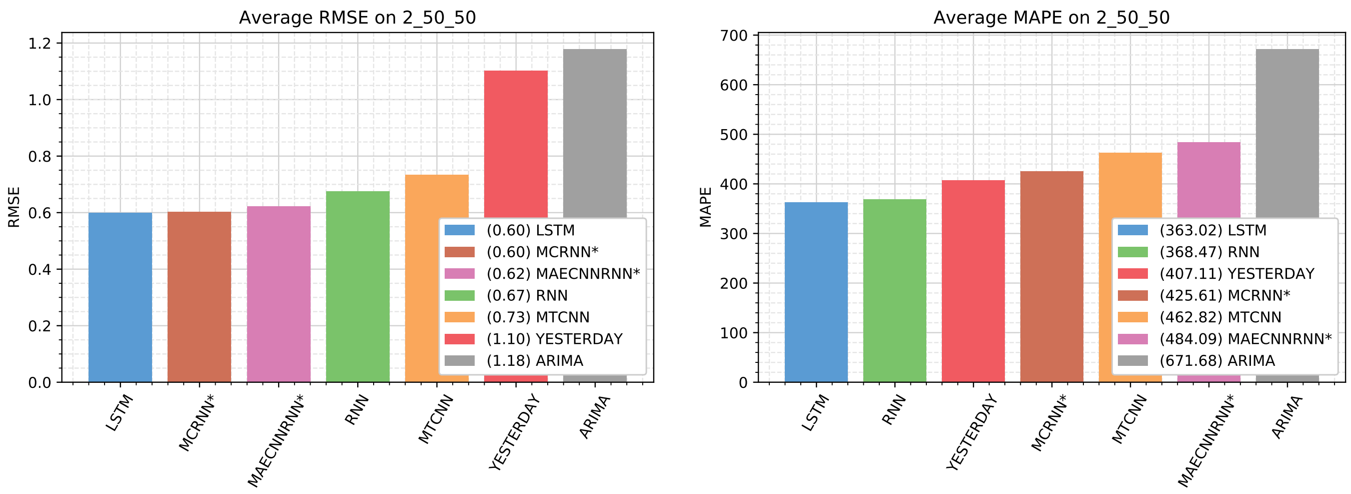

Fig. 17: Average RMSE (left) and MAPE (right) on Google Trends data set for configuration 2_50_50

References

[1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[2] A. Yadav and K. Sahu, "Wind forecasting using artificial neural networks: A survey and taxonomy," *International Journal of Research In Science & Engineering*, vol. 3, 2017.

[3] B. Duraisamy, J. V. Shanmugam, and J. Annamalai, "Alzheimer disease detection from structural mr images using fcm based weighted probabilistic neural network," *Brain Imaging and Behavior*, Feb 2018. [Online]. Available: https://doi.org/10.1007/s11682-018-9831-2

[4] L. Ozyilmaz and T. Yildirim, "Diagnosis of thyroid disease using artificial neural network methods," in *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, vol. 4. IEEE, 2002, pp. 2033–2036.

[5] B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural networks: Applications in industry, business and science," *Commun. ACM*, vol. 37, no. 3, pp. 93–105, Mar. 1994. [Online]. Available: http://doi.acm.org/10.1145/175247.175257

[6] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 240–254, Mar 1994.

[7] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[8] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Management Science*, vol. 6, no. 3, pp. 324–342, 1960. [Online]. Available: https://doi.org/10.1287/mnsc.6.3.324

[9] G. Janacek, "Time series analysis forecasting and control," *Journal of Time Series Analysis*, vol. 31, no. 4, pp. 303–303, 2010.

[10] Z. Tang, C. de Almeida, and P. A. Fishwick, "Time series forecasting using neural networks vs. box- jenkins methodology," *SIMULATION*, vol. 57, no. 5, pp. 303–310, 1991. [Online]. Available: https://doi.org/10.1177/003754979105700508

[11] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[12] D. Ö. Faruk, "A hybrid neural network and arima model for water quality time series prediction," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 586–594, 2010.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] 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," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 802–810. [Online]. Available: http://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf

[15] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *International Conference on Machine Learning*, 2017.

[16] M. Assaad, R. Bon, and H. Cardot, "A new boosting algorithm for improved time-series forecasting with recurrent neural networks," *Information Fusion*, vol. 9, no. 1, pp. 41 – 55, 2008, special Issue on Applications of Ensemble Methods. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253506000820

[17] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *CoRR*, vol. abs/1506.04214, 2015. [Online]. Available: http://arxiv.org/abs/1506.04214

[18] N. Pang, F. Yin, X. Zhang, and X. Zhao, "A robust approach for multivariate time series forecasting," in *Proceedings of the Eighth International Symposium on Information and Communication Technology*, ser. SoICT 2017. New York, NY, USA: ACM, 2017, pp. 106–113. [Online]. Available: http://doi.acm.org/10.1145/3155133.3155172

[19] C. Kongcharoen and T. Kruangpradit, "Autoregressive integrated moving average with explanatory variable (arimax) model for thailand export," in *33rd International Symposium on Forecasting, South Korea*, 2013, pp. 1–8.

[20] M. Palomares and D. Pauly, "A multiple regression model for prediction the food consumption of marine fish populations," *Marine and Freshwater Research*, vol. 40, no. 3, pp. 259–273, 1989.

[21] F. Azam, "Biologically inspired modular neural networks," Ph.D. dissertation, Virginia Tech, 2000.

[22] K. L. Wagstaff, Y. Lu, A. Stanboli, K. Grimes, T. Gowda, and J. Padams, "Deep mars: Cnn classification of mars imagery for the pds imaging atlas," 2018.

[23] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov, "Exploiting image-trained CNN architectures for unconstrained video classification," *CoRR*, vol. abs/1503.04144, 2015. [Online]. Available: http://arxiv.org/abs/1503.04144

[24] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 131–135.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[26] B. K. Yi, N. D. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris, "Online data mining for co-evolving time sequences," in *Proceedings of 16th International Conference on Data Engineering (Cat. No.00CB37073)*, 2000, pp. 13–22.

[27] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.