Transductive LSTM for Time-Series Prediction: an Application to Weather Forecasting

Zahra Karevan, Johan A. K. Suykens *Fellow, IEEE*

***Abstract*—Long Short-Term Memory (LSTM) has shown significant performance on many real-world applications due to its ability to capture long-term dependencies. In this paper, we utilize LSTM to obtain a data-driven forecasting model for an application of weather forecasting. Moreover, we propose Transductive LSTM (T-LSTM) which exploits the local information in time-series prediction. In transductive learning, the samples in the test point vicinity are considered to have higher impact on fitting the model. In this study, a quadratic cost function is considered for the regression problem. Localizing the objective function is done by considering a weighted quadratic cost function at which point the samples in the neighborhood of the test point have larger weights. We investigate two weighting schemes based on the cosine similarity between the training samples and the test point. In order to assess the performance of the proposed method in different weather conditions, the experiments are conducted on two different time periods of a year. The results show that T-LSTM results in better performance in the prediction task.**

***Index Terms*—Transductive Learning, Long Short-Term Memory, Weather Forecasting.**

I. Introduction

Recurrent Neural Networks (RNNs) [1] are one of the most popular data-driven approaches used for time-series prediction. Having information in a sequential form, the input of the RNN in each time step includes the information of the corresponding time step in the sequence and the previous information provided by a feedback loop. This feedback loop makes the output of RNN in the previous time step to be seen as an input in the current one. Despite the power of RNNs to understand the short-term dependencies, they have problems in capturing long-term ones due to the vanishing gradient problem [2]. The vanishing gradient problem expresses that using a gradient-based optimization in RNNs, the error gradients vanish in the earlier inputs after propagating through several steps. Consequently, the information of earlier steps are neglected in prediction or classification task. To avoid the vanishing gradients problem in RNNs, Hochreiter & Schmidhuber proposed Long Short-Term Memory (LSTM) [3] which is able to capture long-term dependencies by utilizing a memory cell and gating method. LSTMs have been widely used and shown significant performance on different sequence learning problems such as video classification, machine translation and speech recognition [4], [5], [6]. Recently, there has been an increasing interest towards using LSTMs for time-series prediction such as air pollution forecasting, diagnosing based clinical data and traffic flow prediction [7], [8], [9].

Mostly, learning methods are based on inductive learning, where it is assumed that there is a function that maps the input to the output and the aim is to approximate the function in an optimal way. Learning based on induction indicates a global model based on all training data points and the model can be used to predict or classify an unseen data point regardless of its position in the feature space. The idea of locality was first introduced by Vapnik in [10] where he discussed the trade off between the capacity of learning and the number of data points. Due to the possible differences in distribution of data in various regions of the feature space, it is meaningful to consider different learning capacities and properties for different areas in the feature space [11].

In transductive learning (also known as local learning), the learning objective is more focused on obtaining good performance in the vicinity of the unseen data point. Note that to classify or predict an unseen test point, the label is assumed to be unknown while the feature vector is known. Based on the idea of transductive learning, the models are created such that their performance in the neighborhood of the test point in the feature space is optimized [12]. One intuitive approach is a weighting scheme that considers weights for the data points in the training set based on their distance from the test point: the data points in the neighborhood of the test point obtain higher weights. As shown in [11], these weights can be binary or real. If the weights are binary, one may utilize a k-Nearest Neighbor (k-NN) approach to select part of the data points and create the model based on these samples. Nevertheless, in case of real values, each data point receives a weight based on its similarity with the test point. The closest sample gets the highest weight and the furthest one receives the lowest one. Such weighting scheme was used in the framework of Moving Least Squares by Levin [13].

Transductive learning has been used in different types of problems. Do & Poulet proposed a parallel ensemble learning algorithm of random local Support Vector Machines which can help in dealing with some of the issues related to big data problems [14]. Gilardi & Bengio utilized transductive learning for spatial data analysis [15]. Zhai et al. deployed transductive learning in a multi-view approach [16]. In several applications, transductive learning outperforms inductive learning in terms of accuracy [17], [18], [19].

The weather system is known as a challenging complex system, and reliable weather forecasting has been the subject of many studies. State-of-the-art methods utilized Numerical Weather Prediction (NWP) which is a computationally intense method and demands thousands of processing units to have reliable forecasting models [20]. Recently, to avoid the shortcomings of NWP, data-driven approaches have become a major interest of the researchers to predict the future behavior of the weather system. In our previous studies, we investigated

transductive learning as a data-driven approach for weather prediction in the frameworks of feature selection and function approximation [21], [22]. The results of these studies suggest that deploying transductive learning can improve the predictions performance.

In this study, we deploy LSTM for temperature prediction and propose Transductive LSTM (T-LSTM) by altering the cost function in the regression problem. In the proposed method, the impact of the training samples on the cost function is determined by their similarity to the test point. In this study, the weights of the training data points are defined based on their cosine similarity with the test point.

The remaining of this paper is as follows: first, in section II we discuss LSTM architecture and how they can be used for regression problems. Then, in section IV we explain the proposed T-LSTM approach. Afterwards, the experimental results on an application of weather forecasting is shown in section V. Finally, the conclusion is provided.

II. Long Short-Term Memory

The original idea of LSTM was initially proposed by Hochreiter & Schmidhuber [3] and since then different approaches have been proposed to improve the performance of LSTM [23], [24], [25], [26]. In this study, we describe and deploy the popular architecture proposed by Gers et al. [27] and used in many other studies such as [28], [29], [6].

LSTM utilizes a gating mechanism to control the information that has to be kept over time, the duration it has to be kept and the time that it can be read through the memory cell [30]. In this paper we consider the LSTM cell as described in the paper of Graves [29]. To process the information, LSTM benefits from three gates. Assuming i_t, f_t, o_t, c_t and h_t to indicate the values of the input gate, forget gate, output gate, memory cell and hidden state at time t in the sequence respectively and x_t be the input of the system at time t, the architecture of the LSTM cell can be defined as follows [29]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (4)$$

$$h_t = o_t \odot \tanh(c_t). \quad (5)$$

Note that $\sigma(\cdot)$ represents the sigmoid as the activation function. Both logistic sigmoid and hyperbolic tangent are applied element-wise. The full weight matrices W_{xj} for $j \in \{i, f, o, c\}$ are the weights for the input x_t in input, forget and output gates and the memory cell. The weight matrices W_{cj} for $j \in \{i, f, o\}$ are diagonal matrices that are related to the connection of the cell memory to different gates. Note that the number of neurons for all gates is a predefined parameter and the equations (1) to (5) are applied for each neuron. Denoting n as the number of neurons, then $\{i_t, f_t, c_t, o_t, h_t\} \in \mathbb{R}^{n\times 1}$. For simplicity, in the rest of the paper to refer to the weights and biases in the LSTM model, we use column vectors w_{lstm} and b_{lstm} which include all the elements in $\{W_{xi}, W_{ci}, W_{xf}, W_{hf}, W_{cf}, W_{xc}, W_{hc}, W_{xo}, W_{ho}, W_{co}\}$ and $\{b_i, d_f, b_c, b_o\}$ respectively. The LSTM equations can be briefly written as follows

$$\begin{cases} c_t = f(c_{t-1}, h_{t-1}, x_t; w_{\text{lstm}}, b_{\text{lstm}}) \\ h_t = g(h_{t-1}, c_{t-1}, x_t; w_{\text{lstm}}, b_{\text{lstm}}) \end{cases} \quad (6)$$

where $g(\cdot)$ and $f(\cdot)$ can be derived from equations (1) to (5).

III. Training LSTM for Time-Series Prediction

Given a uni-variate or a multi-variate time-series, one may consider the whole time-series as a sample and train the LSTM network. For systems with complex dynamics, such as chaotic systems, it was shown by Suykens et al. [31] how one can improve trajectory learning by training on packets of increasing size. In order to improve the learning process and performance, we define smaller sub-samples from the original time-series. As shown in Fig. 1, we deploy a moving window to sample from the original time-series. Note that the illustrations are done for a uni-variate time-series, and it can be extended to a multi-variate case.

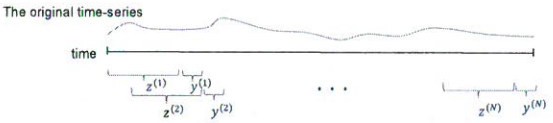

Fig. 1: Sampling from the original time-series

Considering T the sequence length, the sequence $z^{(t)} = [x_t; x_{t+1}; \cdots; x_{t+T-1}] \in \mathbb{R}^T$ and $y^{(t)} = x_{t+T-1+q} \in \mathbb{R}$ are the t-th input and output of the LSTM network, respectively. Note that q is a positive integer that indicates the number of steps ahead to be predicted. Also, N is the total number of subsamples and it depends on the length of the original time-series and the sequence length T.

Fig. 2 depicts how LSTM models process sequential data. The values of gates, memory cell and hidden states are updated in a recurrent way. As shown in the unfolded scheme, in each time step, the input of the LSTM cell includes the input of the corresponding time step and the previous hidden state values.

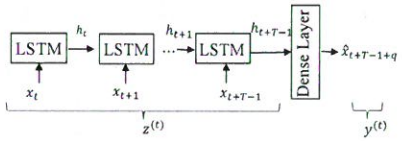

Fig. 2: Unfolded LSTM model

To predict the future behavior of the system, one may use a dense layer as follows

$$\hat{y}^{(t)} = w_{\text{dense}}^T h_{t+T-1} + b_{\text{dense}}, \quad t = 1, \cdots, N, \quad (7)$$

where $w_{\text{dense}} \in \mathbb{R}^{n\times 1}$ and $b_{\text{dense}} \in \mathbb{R}$ are the weights and bias term in the dense layer. Note that the dimension of the hidden state is equal to the predefined number of neurons. Clearly, in the last step, the hidden state values include the information of all previous inputs as it is a function of cell memory and the output gate (5).

In this paper, we use a quadratic loss function to train the network for the regression problem and deploy L_2-norm regularization to avoid overfitting. By denoting w and b all the parameters of the LSTM layer (w_{lstm} and b_{lstm}) and the dense layer (w_{dense} and b_{dense}), the objective function is defined as follow

$$\begin{aligned}(\hat{w}_{\text{lstm}}, \hat{w}_{\text{dense}}, \hat{b}_{\text{lstm}}, \hat{b}_{\text{dense}}) = (\hat{w}, \hat{b}) = \arg\min_{w,b} J \\ J = \frac{1}{N}\sum_{t=1}^{N} (\hat{y}^{(t)} - y^{(t)})^2 + \gamma w^T w\end{aligned} \tag{8}$$

where $\gamma \in \mathbb{R}^+$ is the regularization parameter that has to be tuned and $\hat{y}^{(t)}$ is the estimated value for the output of the system when $z^{(t)}$ is the input sequence. Given an unseen data point $z^{(\eta)}$ with length T and $z_{t'}^{(\eta)}$ representing the t'-th element in the sequence, the hidden state values can be evaluated as follows

$$h_{t'} = g(h_{t'-1}, c_{t'-1}, z_{t'}^{(\eta)}; \hat{w}_{\text{lstm}}, \hat{b}_{\text{lstm}}), \tag{9}$$

where $t' = \eta, \cdots, \eta + T - 1$. Then, the prediction is done as follows

$$\hat{y}^{(\eta)} = \hat{w}_{\text{dense}}^T h_{\eta+T-1} + \hat{b}_{\text{dense}}. \tag{10}$$

IV. Transductive LSTM for Time Series Prediction

A. Optimization problem

Mostly, LSTM is deployed for regression as explained in the previous section. In the corresponding model, all subsamples have the same impact on the model parameters (w, b). However, in the proposed *Transductive LSTM (T-LSTM)*, the influence of each training data point on the parameters depends on the similarity of the test data point, which is a new sequence with length T, to the corresponding data point. Therefore, the emphasis of the training phase is to obtain a better performance in the vicinity of the test point, and higher importance is given to the performance of the model for estimating the training samples that are in the neighborhood of the test point. As a result, all model parameters depend on the test point.

Assuming $z^{(\eta)}$ being an unseen sequence, the state space model of the T-LSTM can be written as follows

$$\begin{cases} c_{t,\eta} = f(c_{t-1,\eta}, h_{t-1,\eta}, x_t; w_{\text{lstm},\eta}, b_{\text{lstm},\eta}) \\ h_{t,\eta} = g(h_{t-1,\eta}, c_{t-1,\eta}, x_t; w_{\text{lstm},\eta}, b_{\text{lstm},\eta}). \end{cases} \tag{11}$$

We use η as a subscript is to indicate that the model parameters depend on the new given data point $z^{(\eta)}$. Afterwards, the prediction using the dense layer can be written as follows

$$\hat{y}_\eta^{(t)} = w_{\text{dense},\eta}^T h_{t+T-1,\eta} + b_{\text{dense},\eta}, \quad t = 1, \cdots, N, \tag{12}$$

where $w_{\text{dense},\eta} \in \mathbb{R}^{n\times 1}$ and $b_{\text{dense},\eta} \in \mathbb{R}$ are the weights and bias term in the dense layer.

To specify the dependencies on the new unseen data point, assume $s_{t,\eta}$ being the similarity between length T sequences $z^{(t)}$ and $z^{(\eta)}$, and w_η and b_η denoting all the parameters in $(w_{\text{lstm},\eta}, w_{\text{dense},\eta})$ and $(b_{\text{lstm},\eta}, b_{\text{dense},\eta})$ respectively. The objective function can be written as follows

$$\begin{aligned}(\hat{w}_{\text{lstm},\eta}, \hat{w}_{\text{dense},\eta}, \hat{b}_{\text{lstm},\eta}, \hat{b}_{\text{dense},\eta}) = (\hat{w}_\eta, \hat{b}_\eta) = \arg\min_{w_\eta, b_\eta} J_\eta \\ J_\eta = \frac{1}{N}\sum_{t=1}^{N} s_{t,\eta}(\hat{y}_\eta^{(t)} - y^{(t)})^2 + \gamma_\eta w_\eta^T w_\eta\end{aligned} \tag{13}$$

where $s_{t,\eta} \in \mathbb{R}^+$. Note that the main difference between (8) and (13) is the fact that in the latter, all the model parameters depend on the new point $z^{(\eta)}$. Thus, for each unseen subsample the model has to be retrained. This means that all the parameters $\hat{w}_\eta$ and $\hat{b}_\eta$ can be different for each test point. Therefore, the main drawback of the proposed method is its high time complexity.

The similarity function $s_{t,\eta}$ can take binary or positive real values. In case of a binary value, k-Nearest Neighbors of $z^{(\eta)}$ are selected as training data and the rest of the samples are discarded. In this case, $s_{t,\eta}$ for all selected samples are equal to one. Hence, their impact on the training phase is the same. In addition, one may deploy a clustering approach to select the samples similar to the test point. Bandara et al. [30] utilized clustering to group the data and then trained a separate LSTM model per cluster. Note that in this case the number of data points to train the model is reduced.

On the other hand, in case $s_{t,\eta}$ are real-valued, all subsamples are taken into account in the training phase, while their impact on the training phase is different. In this case $s_{t,\eta}$ can be calculated using a similarity function, e.g. Gaussian or cosine similarity. In this study we use cosine similarity to weight the training samples. Cosine similarity is defined as $\kappa_{t,\eta} = \frac{z^{(t)^T} z^{(\eta)}}{||z^{(t)}||||z^{(\eta)}||}$ and it takes a value between -1 and 1. To achieve a non-negative similarity value, we perform two transformations on the cosine similarity: 1) linear transformation at which point $s_{t,\eta} = \kappa_{t,\eta} + 1$ such that the similarity is between 0 and 2, and 2) non-linear sigmoid transformation at which point $s_{t,\eta} = \sigma(\kappa_{t,\eta}) = \frac{1}{1+\exp(-\kappa_{t,\eta})}$ such that the similarity is between 0 and 1. Note that, other similarity functions such as the Gaussian similarity can be used to specify the weights; however, in this study we deploy cosine similarity to avoid additional tuning parameters.

For a given point $z^{(\eta)}$, the updates of the hidden state are done as follows

$$h_{t',\eta} = g(h_{t'-1,\eta}, c_{t'-1,\eta}, z_{t'}^{(\eta)}; \hat{w}_{\text{lstm},\eta}, \hat{b}_{\text{lstm},\eta}) \tag{14}$$

where $t' = \eta, \cdots, \eta + T - 1$. Afterwards, the final prediction can be obtained as follows

$$\hat{y}_\eta^{(\eta)} = \hat{w}_{\text{dense},\eta}^T h_{\eta+T-1,\eta} + \hat{b}_{\text{dense},\eta}. \tag{15}$$

Note that (9), (10), (14) and (15) refer to the prediction of the system when $z^{(\eta)}$ is the input. However, the hidden state updates and the prediction model in the T-LSTM depend also implicitly on the new point $z^{(\eta)}$ as the model parameters are optimized based on the similarity between the training points

and $z^{(\eta)}$. This is indicated by η subscript in $h_{t',\eta}$ and $\hat{y}_{\eta}^{(\eta)}$ in (14) and (15).

B. Tuning parameters

Similar to (8), γ_η is a tuning parameter in (13). In addition, the number of neurons in LSTM gates is a tuning parameter both in inductive and transductive approaches.

Since the emphasis of the T-LSTM is to achieve a good performance in the vicinity of the test point, one may select the most similar samples to the test point as the validation set. However, in this scenario the model cannot exploit the information of most similar data points in the training phase. In this study, we assume that there is no sudden change in the pattern of the original series; thus, the distance between two consecutive samples in the feature space is not large. Therefore, the last samples of the training period, which include the samples prior to the test set, should be in the neighborhood of the test point in the feature space. Hence, these samples are selected as the validation set.

V. Experiments

A. Dataset

In this study, data have been collected from the Weather Underground website and include real measurements of weather variables such as minimum and maximum temperature, dew point, and wind speed for 5 cities including Brussels, Antwerp, Liege, Amsterdam and Eindhoven, as shown in Fig. 3.

Fig. 3: Weather stations (picture from Google maps)

The data cover a time period from the beginning of 2007 to mid-2014 and contain 18 measured weather variables for each day per city. To exploit all available information, the number of training data points is different for each test set, and it is equal to the number of days from the beginning of 2007 until the day before the test set.

To evaluate the performance of the proposed methods in various weather conditions, two test sets are defined: (i) from mid-November 2013 to mid-December 2013 (Nov/Dec) and (ii) from mid-April 2014 to mid-May 2014 (Apr/May).

B. Model Selection

As the difference between weather variables in two days in a row is not significant, in this study we use the last samples of the training period as validation set. For each test set, two months prior to the test set is considered as the validation set. Hence, for the Nov/Dec test set, the validation set includes the period from mid-September 2013 to mid-November 2013 and for the Apr/May test set, it includes the period from mid-February 2014 to mid-April 2014. The samples that are left for training the models are shuffled.

In this study the implementation of the LSTM models are in TensorFlow, which is based on the model described by Zaremba & Vinyals [28]. Note that this model is slightly different from the one discussed by Graves [29]. The tuning parameters are the number of neurons and the regularization parameter. For the number of neurons in the LSTM, we try the values: $\{16, 32, 64, 128, 256\}$, and for the regularization parameter γ the values $\{0.1, 0.2, 1, 2, 10\}$ are considered. For the inner state, mostly tanh is used as the activation function. In addition to that, in section V-C we discuss the performance while sigmoid is chosen as the activation function. The ADAM optimizer is used in this paper to optimize the objective function.

C. Results

In Fig. 4 the cosine similarity of the training data points to the prototypes of both tests sets is depicted. As can be seen the similarity values have a temporal meaning; i.e. the samples in a specific period of the year are similar. In addition, it can be concluded that there are no significant changes in the weather variables for two temporally close samples.

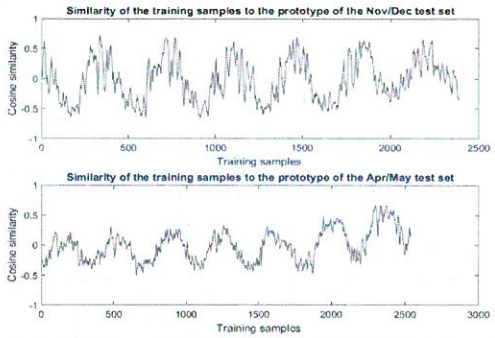

Fig. 4: Similarity of the training samples to both test sets

The performance of the proposed method is evaluated on an application of maximum and minimum temperature prediction in Brussels, Belgium. The forecasting is done for one to six days ahead. To avoid local minima problems in LSTMs, we did the experiments five times. Tables I and II present the median Mean Absolute Error (MAE) and Mean Squared Error (MSE) of the inductive and transductive approaches on the test sets for different internal activation function and sequence length. The bold one shows the best performance for the corresponding setting. The results for the transductive approaches are presented for both linear ($\kappa_{t,\eta} + 1$) and non-linear ($\sigma(\kappa_{t,\eta})$) transformation of the cosine similarity.

As is shown, the transductive approaches outperform the inductive approach in the case that the transfer function is a sigmoid. In case of a hyperbolic tangent transfer function, there are a few cases that the inductive approach has better performance than the transductive approach. In addition, there is no significant difference between the linear and non-linear transformation of the cosine distance.

The comparison between the MAE of the best models of inductive and transductive approaches is made in Figures 5

TABLE I: MAE and MSE of minimum and maximum temperature prediction for LSTM, T-LSTM with both linear and non-linear transformation of cosine similarity, for sequence length 10 and 20, and for 1 to 6 days ahead for Nov/Dec test set

			Activation function: *Tanh*						Activation function: *Sigmoid*					
			Mean Absolute Error			Mean Squared Error			Mean Absolute Error			Mean Squared Error		
Seq. Len.	Steps ahead	Temp.	LSTM	T-LSTM ($\kappa+1$)	T-LSTM ($\sigma(\kappa)$)	LSTM	T-LSTM ($\kappa+1$)	T-LSTM ($\sigma(\kappa)$)	LSTM	T-LSTM ($\kappa+1$)	T-LSTM ($\sigma(\kappa)$)	LSTM	T-LSTM ($\kappa+1$)	T-LSTM ($\sigma(\kappa)$)
10	1	Min	1.54	1.54	**1.50**	3.84	**3.74**	**3.74**	1.72	**1.61**	1.66	4.61	**4.10**	4.20
		Max	1.88	1.32	**1.30**	4.58	2.92	**2.82**	1.52	1.41	**1.37**	3.80	**3.49**	3.52
	2	Min	**1.74**	1.83	**1.74**	**4.32**	4.70	4.42	1.89	**1.75**	1.76	5.48	**4.37**	5.35
		Max	2.16	**1.88**	1.94	6.47	5.13	**5.10**	1.61	**1.55**	1.59	4.41	**3.87**	3.99
	3	Min	**1.74**	1.77	**1.74**	4.40	4.46	**4.35**	2.08	1.95	**1.91**	6.97	5.35	**5.29**
		Max	**1.95**	1.99	**1.95**	**4.73**	4.95	5.19	2.10	**1.79**	1.81	7.51	6.26	**5.90**
	4	Min	**1.56**	1.67	1.57	**3.73**	4.31	3.80	2.02	1.85	**1.70**	6.36	5.50	**4.60**
		Max	1.94	**1.59**	1.86	4.45	**3.96**	4.34	1.97	**1.90**	1.92	6.46	**5.79**	5.89
	5	Min	**1.67**	1.81	**1.67**	4.47	4.90	**4.30**	2.01	1.92	**1.76**	5.84	5.40	**4.47**
		Max	1.95	1.86	**1.59**	5.44	4.42	**3.93**	1.89	**1.81**	1.91	5.79	**5.40**	5.99
	6	Min	1.96	2.22	**1.87**	5.81	**5.18**	5.25	2.02	1.90	**1.73**	5.50	4.80	**4.28**
		Max	1.90	1.83	**1.76**	5.98	5.90	**5.13**	2.09	**1.73**	1.83	6.96	**5.24**	5.49
20	1	Min	1.76	1.58	**1.49**	5.40	4.07	**3.67**	1.74	**1.65**	1.69	4.53	**4.19**	4.40
		Max	1.48	1.39	**1.34**	3.32	3.10	**2.85**	1.43	1.43	**1.35**	3.49	3.59	**3.38**
	2	Min	1.77	1.80	**1.71**	4.40	4.51	**4.28**	1.85	**1.74**	**1.74**	5.63	5.06	**4.92**
		Max	1.83	1.74	**1.51**	4.55	**4.09**	4.26	1.60	**1.48**	1.49	4.15	**3.53**	3.62
	3	Min	1.77	1.80	**1.75**	**4.46**	4.61	4.60	2.03	1.96	**1.94**	6.26	5.55	**5.43**
		Max	**1.73**	2.09	1.84	**3.90**	5.60	4.51	1.92	**1.73**	1.76	5.67	5.76	**5.57**
	4	Min	1.62	1.62	**1.60**	3.92	**3.68**	3.86	1.97	**1.69**	1.72	6.04	**4.50**	4.74
		Max	1.99	1.99	**1.91**	4.70	5.04	**4.43**	1.94	1.86	**1.80**	4.98	5.10	**4.65**
	5	Min	**1.68**	**1.68**	1.69	4.48	4.38	**4.31**	2.00	**1.68**	1.76	5.75	**4.81**	4.78
		Max	**1.54**	2.09	1.57	**3.37**	5.65	3.68	2.01	**1.75**	1.76	4.99	**4.25**	4.30
	6	Min	**1.85**	1.87	1.90	**5.28**	5.31	5.42	1.98	**1.76**	1.77	5.30	4.46	**4.28**
		Max	1.85	1.79	**1.76**	6.94	6.26	**5.46**	1.94	1.77	**1.71**	6.19	5.40	**5.22**

TABLE II: MAE and MSE of minimum and maximum temperature prediction for LSTM, T-LSTM with both linear and non-linear transformation of cosine similarity, for sequence length 10 and 20, and for 1 to 6 days ahead for Apr/May test set

			Activation function: *Tanh*						Activation function: *Sigmoid*					
			Mean Absolute Error			Mean Squared Error			Mean Absolute Error			Mean Squared Error		
Seq. Len.	Steps ahead	Temp.	LSTM	T-LSTM ($\kappa+1$)	T-LSTM ($\sigma(\kappa)$)	LSTM	T-LSTM ($\kappa+1$)	T-LSTM ($\sigma(\kappa)$)	LSTM	T-LSTM ($\kappa+1$)	T-LSTM ($\sigma(\kappa)$)	LSTM	T-LSTM ($\kappa+1$)	T-LSTM ($\sigma(\kappa)$)
10	1	Min	1.74	**1.52**	1.68	4.08	**3.94**	4.41	1.63	1.59	**1.56**	4.04	**3.73**	3.88
		Max	2.15	2.10	**2.04**	**7.44**	6.78	**7.44**	2.21	**2.12**	2.16	7.14	**7.07**	7.36
	2	Min	2.05	1.99	**1.95**	**7.42**	7.69	8.86	2.33	1.96	**1.91**	7.95	7.30	**7.02**
		Max	2.33	2.23	**2.19**	8.24	8.16	**8.09**	2.68	**2.33**	2.47	9.91	**8.23**	8.74
	3	Min	2.07	**1.98**	2.02	8.54	7.59	**7.54**	2.08	1.96	**1.95**	7.22	**7.07**	7.22
		Max	2.71	**2.15**	2.17	12.77	**7.30**	7.50	2.55	2.37	**2.36**	9.10	**7.89**	8.20
	4	Min	2.16	**1.93**	2.14	8.47	**6.42**	6.60	2.07	**1.92**	2.19	6.63	**6.29**	7.04
		Max	2.87	**2.22**	**2.22**	13.41	7.97	**7.86**	2.36	2.30	**2.25**	9.04	8.65	**7.96**
	5	Min	2.05	2.02	**2.01**	**6.91**	7.17	7.30	2.35	**2.11**	2.22	7.84	**6.94**	7.18
		Max	2.85	2.52	**2.26**	13.31	10.59	**8.32**	2.53	2.28	**2.19**	10.59	8.86	**8.33**
	6	Min	2.66	**2.20**	2.28	13.80	8.35	**7.48**	2.52	**2.35**	2.54	8.58	**7.95**	8.61
		Max	2.95	**2.58**	2.72	14.40	**11.38**	12.70	2.44	**2.36**	2.43	10.38	**8.54**	10.38
20	1	Min	1.62	**1.52**	1.58	4.25	**4.08**	4.42	1.57	**1.50**	1.67	4.03	**3.80**	4.03
		Max	2.20	**2.10**	2.13	7.60	**7.03**	7.33	2.16	**2.11**	2.13	7.56	**7.43**	7.89
	2	Min	2.00	1.92	**1.91**	7.54	7.90	**7.17**	2.03	1.94	**1.90**	7.95	**7.38**	7.56
		Max	2.38	2.31	**2.25**	8.33	**7.81**	8.00	2.61	2.57	**2.50**	9.66	9.36	**8.99**
	3	Min	2.17	**1.93**	2.05	8.32	**6.21**	7.54	2.00	**1.97**	1.98	8.39	7.77	**7.39**
		Max	2.66	2.39	**2.20**	9.50	8.85	**7.37**	2.33	**2.23**	2.33	7.72	**7.30**	7.73
	4	Min	2.21	2.14	**2.06**	7.58	7.05	**6 .37**	2.18	2.02	**1.95**	7.10	**6.41**	6.92
		Max	2.88	**2.39**	2.49	13.44	**8.85**	8.88	2.22	**2.15**	2.20	7.48	**7.35**	7.56
	5	Min	2.37	2.23	**2.18**	9.70	9.40	**6.94**	2.28	**2.05**	2.32	7.53	**7.32**	7.65
		Max	2.94	**2.52**	2.78	12.26	10.57	**10.40**	2.73	2.54	**2.30**	12.49	9.01	**8.64**
	6	Min	2.37	**2.28**	**2.28**	9.65	**8.10**	9.70	2.45	**2.31**	2.43	8.28	**7.84**	8.15
		Max	2.76	2.67	**2.62**	10.61	**9.79**	9.82	2.79	2.51	**2.23**	11.12	10.73	**8.30**

and 6. In many cases the transductive approach outperforms the inductive approach or provides equally good performance. In the Apr/May test set in all cases the transductive approach shows better performance than the global one, whereas in the Nov/Dec test set the inductive and the transductive methods are more competitive.

In addition, in Figures 5 and 6, the performance of the LSTM models, as data-driven approaches, are compared with the performance of Weather Underground (www.wunderground.com) which is a well-known company for weather forecasting. The results reveal that while the information that is used in the data-driven approaches is limited, the performance is competitive with the state-of-the-art models used by Weather Underground.

VI. Conclusion

In this study, we deployed an LSTM model for an application of temperature prediction. We proposed the T-LSTM model to exploit transductive learning in time-series prediction. In this case, the model can be achieved by altering the cost function of the LSTM model such that the samples in the neighborhood of the test point have a higher impact on the objective function. We tested two weighting schemes which are based on the cosine similarity between the training data points and the test point. The experiments are done for two different periods in a year to evaluate the performance of the proposed method in different weather conditions. We

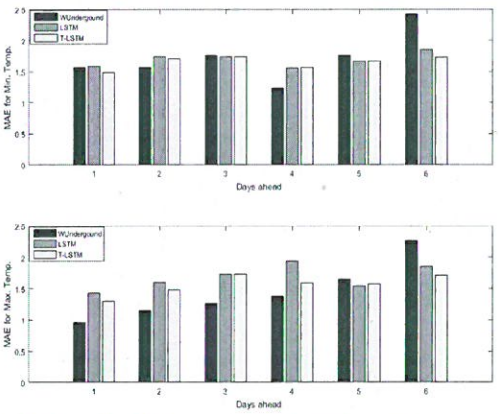

Fig. 5: Comparing MAE of minimum and maximum temperature prediction for Weather Underground, LSTM, and T-LSTM for 1 to 6 days ahead for Nov/Dec test set

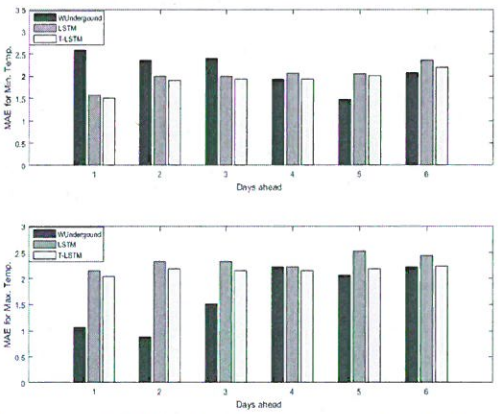

Fig. 6: Comparing MAE of minimum and maximum temperature prediction for Weather Underground, LSTM, and T-LSTM for 1 to 6 days ahead for Apr/May test set

examined different sequence length and transfer functions and the results suggest that transductive LSTM outperforms LSTM in many cases. Moreover, it is shown that the performance of the inductive and transductive LSTM models as data-driven approaches are competitive with the state-of the-art methods in weather forecasting.

ACKNOWLEDGMENT

Research supported by Research Council KUL: CoE PFV/10/002 (OPTEC), PhD/Postdoc grants Flemish Government; FWO: projects: G0A4917N (Deep restricted kernel machines), G.088114N (Tensor based data similarity), ERC Advanced Grant E-DUALITY (787960).

REFERENCES

[1] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

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

[4] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *CVPR, 2015*. IEEE, 2015, pp. 4694–4702.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[6] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*. IEEE, 2013, pp. 6645–6649.

[7] B. S. Freeman, G. Taylor, B. Gharabaghi, and J. Thé, "Forecasting air quality time series using deep learning," *Journal of the Air & Waste Management Association*, no. just-accepted, 2018.

[8] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, "Learning to diagnose with LSTM recurrent neural networks," *ICLR*, 2016.

[9] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on*. IEEE, 2015, pp. 153–158.

[10] V. Vapnik, "Principles of risk minimization for learning theory," in *Advances in neural information processing systems*, 1992, pp. 831–838.

[11] L. Bottou and V. Vapnik, "Local learning algorithms," *Neural Computation*, vol. 4, no. 6, pp. 888–900, 1992.

[12] S. Pang and N. Kasabov, "Inductive vs transductive inference, global vs local models: SVM, TSVM, and SVMT for gene expression classification problems," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2. IEEE, 2004, pp. 1197–1202.

[13] D. Levin, "The approximation power of moving least-squares," *Mathematics of Computation of the American Mathematical Society*, vol. 67, no. 224, pp. 1517–1531, 1998.

[14] T.-N. Do and F. Poulet, "Random local SVMs for classifying large datasets," in *International Conference on Future Data and Security Engineering*. Springer, 2015, pp. 3–15.

[15] N. Gilardi and S. Bengio, "Local machine learning models for spatial data analysis," *Journal of Geographic Information and Decision Analysis*, vol. 4, no. EPFL-ARTICLE-82651, pp. 11–28, 2000.

[16] D. Zhai, X. Liu, H. Chang, Y. Zhen, X. Chen, M. Guo, and W. Gao, "Parametric local multiview Hamming distance metric learning," *Pattern Recognition*, vol. 75, pp. 250–262, 2018.

[17] E. E. Elattar, J. Goulermas, and Q. H. Wu, "Electric load forecasting based on locally weighted support vector regression," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 4, pp. 438–447, 2010.

[18] L. Chen, M. Zhou, M. Wu, J. She, Z. Liu, F. Dong, and K. Hirota, "Three-layer weighted fuzzy support vector regression for emotional intention understanding in human-robot interaction," *IEEE Transactions on Fuzzy Systems*, 2018.

[19] Q. Wu, H. Li, F. Meng, and K. N. Ngan, "Generic proposal evaluator: A lazy learning strategy toward blind proposal quality assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 306–319, 2018.

[20] P. Bauer, A. Thorpe, and G. Brunet, "The quiet revolution of numerical weather prediction," *Nature*, vol. 525, no. 7567, pp. 47–55, 2015.

[21] Z. Karevan and J. A. K. Suykens, "Transductive feature selection using clustering-based sample entropy for temperature prediction in weather forecasting," *Entropy*, vol. 20, no. 4,264, 2018.

[22] Z. Karevan, Y. Feng, and J. A. K. Suykens, "Moving least squares support vector machines for weather temperature prediction," in *ESANN*, 2016, pp. 611–616.

[23] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[24] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez, "Training recurrent networks by evolino," *Neural Computation*, vol. 19, no. 3, pp. 757–779, 2007.

[25] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.

[26] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *IJCNN*, vol. 3. IEEE, 2000, pp. 189–194.

[27] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[28] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[29] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[30] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using long short-term memory networks on groups of similar series," *arXiv preprint arXiv:1710.03222*, 2017.

[31] J. A. K. Suykens, J. P. L. Vandewalle, and B. L. De Moor, *Artificial neural networks for modelling and control of non-linear systems*. Springer Science & Business Media, 1996.