Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 265–276
DOI: 10.15439/2025F7041
ISSN 2300-5963 ACSIS, Vol. 43

A Comparative Study of LSTM Efficiency vs. Transformer Power for Localized Time Series Forecasting

Twisampati Sarkar
School of Computing and Informatics
University of Louisiana at Lafayette, U.S.A.
Email: twisampati.sarkar1@louisiana.edu
ORCID: 0009-0003-6872-358X

Chee-Hung Henry Chu
School of Computing and Informatics
University of Louisiana at Lafayette, U.S.A.
Email: chu@louisiana.edu
ORCID: 0000-0002-5817-8798

***Abstract*—Forecasting multivariate time series increasingly uses deep learning, including models inspired by Neural Machine Translation (NMT). While Transformers excel at long-range dependencies, their computational overhead may not suit all problems. This study advocates the relevancy of Long Short-Term Memory (LSTM)-based encoder-decoder networks for scenarios with shorter input windows and prediction horizons. We present a comprehensive empirical evaluation of four classical LSTM-based NMT models, including variants with attention mechanisms specifically adapted for multistep time series forecasting. Our assessment focuses on their performance and the impact of varying input window sizes and prediction horizons within these computationally efficient, short-sequence contexts. We empirically compare these LSTM-based models against Transformer baselines operating under the same short input window and prediction horizon constraints.**

Key findings indicate that: (i) LSTM-based NMT models achieve or exceed existing state-of-the-art results for short-term predictions; (ii) within smaller input configurations, input window size minimally affects forecasting performance for tested horizons, suggesting efficiency gains are possible; (iii) for attention-based NMT models, attention scoring critically influences accuracy, demanding careful selection; (iv) our comparative analysis demonstrates that for time series problems where immediate historical context is sufficient, LSTM-based encoder-decoders are competitive with, or even outperform, Transformer-based models while offering a more computationally efficient solution.

Overall, our findings signify that original LSTM-based NMT models are robust and capable tools, particularly well-suited for short-term time series prediction tasks where local pattern capture and computational efficiency are priorities, even in the era of Transformers.

***Index Terms*—Attention Mechanism; Long Short-Term Memory; Transformers; Neural Machine Translation; Time Series Forecasting**

I. Introduction

MULTIVARIATE time series are embedded in our day-to-day activities; examples are health and mobile sensor readings, contagious disease markers, power consumption, road occupancy rates and traffic flow indicators, financial stock prices and currency exchange rates. A primary challenge in forecasting multivariate time series lies in extracting both complex temporal patterns—such as short-term trends and yearly seasonalities—and dynamic, nonlinear interdependencies among the individual driving variables of the multiple series. Different machine learning methods have been used to predict the trends [1], [2] as well as values [3] of the time series. Deep learning-based methods are increasingly utilized for such multivariate, multistep time series analysis tasks, including forecasting energy demand [4], network intrusion [5], and anomaly detection [6], [7]. In particular, deep learning architectures leveraging Recurrent Neural Networks (RNNs) effectively address many shortcomings of traditional models in these analytical tasks [8], [9].

Deep learning models for time series forecasting often draw inspiration from Neural Machine Translation (NMT) frameworks [10], highly successful in Natural Language Processing (NLP) and Computer Vision (CV). NMT models translate word sequences from one language to another, traditionally using an encoder to process the input into a fixed-length vector and a decoder to generate the output sequence [11]. This encoder-decoder paradigm adapts to time series prediction by treating past observations as an "input language" and future predictions as an "output language," effectively "translating" past sequences into future values. Long Short-Term Memory (LSTM) [12] is a specialized type of RNN particularly adept at sequential data tasks. LSTMs can be employed in an NMT encoder-decoder architecture: the encoder LSTM reads the input sequence (source language) word by word, processing it into a fixed-length context vector that encapsulates the entire input's meaning; subsequently, the decoder LSTM takes this context vector as its initial hidden state and generates the output sequence (target language) word by word, leveraging the learned information from the encoder to produce a coherent and accurate translation [13], [14].

Attention mechanisms [15] critically advanced NMT, allowing decoders to selectively focus on relevant parts of the input sequence (or past time series points), significantly improving performance for translation or prediction. LSTM-

based NMT models can incorporate an attention mechanism to overcome the limitation of fixed-length context vectors by allowing the decoder to dynamically "attend" to different parts of the encoder's output at each decoding step, giving more weight to relevant source words when generating a particular target word. Transformer networks, a more recent NMT evolution, move beyond earlier RNN-based models by relying entirely on attention mechanisms to draw relationships between input and output sequence positions.

Traditional statistical approaches to time series forecasting have often focused on predicting a single future time step. However, contemporary applications increasingly demand predictions over a longer horizon—not just one step ahead, but potentially tens or even hundreds of time steps into the future. This extended prediction horizon seems to be well matched to NMT models' capabilities. For example in NLP it is not unusual to translate, say, five words in one language to, say, ten words in a different language. When extending this scenario to time series forecasting, it is natural to ask how the NMT models would perform in predicting a time sequence ("horizon") longer than the input sequence ("input window").

Given the recent advancements and dominance of Transformer-based architectures in handling long sequences, what then is the continued relevance and precise niche for LSTM-based models in time series forecasting, particularly when considering resource constraints or the need to understand specific short-to-medium range forecasting behaviors? To precisely delineate this niche and understand their practical applicability, we study how LSTM-based encoder-decoder models, particularly their performance across varied input and prediction window sizes and the influence of different attention mechanisms, compare in efficiency and effectiveness against contemporary Transformer models when specific short-to-medium range forecasting behaviors are critical. This raises the following questions:

1) How would these models behave when both the input window size and the predicted horizon window size are varied on a narrow time interval?
2) Although [16] presents a comparison of the different attention scoring functions on NLP tasks, how do the scoring functions affect the performance of these models for time series analysis? Does the use of different attention scoring function for different attention-based NMT models as described in [16], [17] have a significant effect on the performance of the attention-based models?
3) While Transformers excel at longer contexts, might LSTMs still offer competitive or superior performance and efficiency for shorter input windows and prediction horizons, a common requirement in many real-world applications?

To address these pertinent questions and delineate opportunities for enhancing time series forecasting, this research undertakes an empirical investigation into the performance of NMT models. Specifically, we evaluate and compare two baseline (or 'vanilla') NMT models [10], [18] with two attention-equipped NMT models [16], [17] and compared them to three Transformer models [19], [20], [21].

Based on the findings from our experimental study on multivariate time series forecasting models, we note the following contributions.

1) We provide a thorough empirical evaluation of classical LSTM-based encoder-decoder networks for multivariate time series forecasting, assessing their performance across various real-world datasets and different input window and prediction horizon configurations.
2) We examine the effect of varying input window sizes, demonstrating that even smaller input contexts can yield accurate predictions for longer horizons in certain time series. Furthermore, we confirm that the choice of attention scoring function significantly impacts forecasting accuracy for attention-based LSTM models, emphasizing the need for careful selection.
3) We empirically compare the performance of LSTM-based encoder-decoder models against Transformer-based architectures when both operate under comparable short input window widths and prediction horizons. Our findings suggest that for these specific local forecasting scenarios, LSTM-based methods can achieve competitive or even superior performance, often with greater computational efficiency, thereby highlighting their continued relevance where long-range dependencies are not the primary drivers or resources are constrained.
4) This study contributes to a more nuanced understanding of deep learning for time series forecasting by delineating specific contexts where LSTM-based models remain a highly viable and efficient choice, thus informing the selection of appropriate models based on problem characteristics (e.g., emphasis on local patterns, computational budget) rather than solely relying on state-of-the-art benchmarks achieved with extensive contexts.

The remainder of this paper is organized as follows. In Section II, we describe traditional and deep learning methods for time series forecasting, discussing their limitations and advantages, and explain the relevance of NMT models, including the recently developed Transformer methods. In Section III, we provide details of the LSTM-based encoder-decoder models used for time series forecasting and explain the model architecture, attention mechanisms, and training procedure. Section IV presents the experimental setup, including datasets, parameter settings, evaluation metrics, and results, along with a discussion and analysis of the findings. Finally, we draw our conclusion and suggest directions for future research in Section V.

II. Related Work

Most time series data in real world scenarios involves a mixture of long and short term patterns. Time series forecast-

ing models therefore have been developed to capture long or short term, or both, recurring patterns for accurate predictions. Classical approaches include traditional statistical methods such as Auto-Regressive (AR) models, e.g. ARIMA [22], Box-Jenkins [23], [24], Moving Averages, e.g. Holt and Winters method [25], [26]. Vector Auto-regression (VAR) family of models such as structured VAR [27] and elliptical VAR [28] are preferred when handling high-dimensional time series as they have the inherent property of AR models and are agnostic to the dependencies between output variables. However, the VAR models tend to overfit when modeling longer time intervals. Nonparametric algorithms such as Gaussian Processes [29] model the complex dynamic temporal relationship with a Bayesian approach, at a higher computational cost. Although these methods are somewhat effective, they fall short in capturing the complicated nonlinear dependencies between a wider or longer interval of observed time series signals and between multiple high-dimensional variables.

A. Recurrent Neural Networks and Encoder-Decoder Models

RNN-based networks [30]—based on LSTM or Gated Recurrent Units (GRU) [12], [18]—were the first sequential deep learners for time series analysis, where they encode past information as a fixed-length vector. An RNN learns a fixed-length representation from multiple sequences of arbitrary length. Advantages of using RNN-based models are as follows. Firstly, it has been shown [31] that RNN fits into the Nonlinear Autoregressive Moving Average framework. The state of the hidden layer in an RNN at any given time is dependent on the previous time steps. Hence RNNs have the inherent desirable properties of AutoRegression and Moving Average-based statistical methods. Secondly, RNNs represent time recursively, making temporal dependencies easier to learn [30]. Third, the recursive property of the RNN model allows it to store complex signals for a variable amount of time. These properties make the RNN a good choice for modeling or learning sequences of variable length data, where each sequence of data points can be assumed to be dependent on previous ones [32]. However, learning in RNNs can suffer from the problems of vanishing and exploding gradients [33], [34] and thus RNNs can have difficulty capturing long-term temporal dependencies. Vanishing gradients are mitigated by using LSTM or GRU units, which can effectively learn long-range temporal dependencies [12], [18]. Both the LSTM and GRU variants of RNN have been the foundations for many state-of-the-art algorithms in speech recognition, machine translation, sentiment analysis and other NLP tasks. LSTMs have also played a significant role in capturing temporal dependencies in CV tasks such as activity classification, detection and other video tasks [35], [36].

A prominent variant of RNN architecture is the encoder-decoder (sequence-to-sequence) model [18], [37]. The encoder-decoder model sequentially links two RNNs—that serve as an encoder and a decoder—through a fixed size vector, generally the last encoder state. The concept is to encode the input sequence as a fixed-length vector representation and use the decoder to translate the learned, encoded fixed-length vector into a variable output sequence. LSTM and GRU-based encoder-decoder networks are popular due to their success in neural machine translation.

B. Attention Mechanisms

Embedding all the information in a fixed-size vector may result in information loss. In addition, the performance of sequence-to-sequence models deteriorates as the length of sequences of the encoder or the decoder increases [38]. To overcome these shortcomings, the attention mechanism is introduced to the architecture to focus on important parts of the input temporal sequences, by adding relevance calculation from all the encoder cells to each decoder cell [17]. This method has been proven to be successful in various NMT tasks in which the translation of each word from one language to another requires specific attention to particular words in the input source sequence [39].

A prominent example of an attention-embedded architecture is one that uses a dual attention mechanism in an LSTM-based encoder-decoder with a two-stage process. The first attention stage automatically extracts essential driving variables from the encoder's previous hidden state, while the second selects the encoder's hidden states across all time steps [40]. This is effectively complemented by self-attention mechanisms integrated with convolutional layers, which further enhance the model's ability to focus on relevant information [41].

Other notable architectures include TreNet, a hybrid network combining LSTM and temporal Convolutional Neural Network (CNN) modules to predict time series trends [42]. Similarly, LSTNet [8] utilizes temporal one-dimensional convolutions for short-term patterns and LSTMs for long-term dependencies. Interestingly, the LSTNet integrates both components with an autoregressive module, demonstrating a significant performance drop when this component is removed. Their research also indicates that attention does not consistently improve results across all datasets, whereas skip connections have proven more effective [8], [43].

While many models focus on point predictions, DeepAR proposes outputting a probability density function based on parameter estimations at each time step [44]. However, this approach relies on assumptions about the time series data's distribution, which may not always be realistic in real-world scenarios.

Further advancements in recurrent architectures for time series forecasting include a bidirectional LSTM-based encoder-decoder that incorporates a position-based attention mechanism to exploit periodic patterns and strong variable correlations within the data [45]. For multi-quantile probabilistic forecasts, an approach utilizing LSTM-based encoders and decoders has been proposed, though primarily tested on univariate datasets [46]. Additionally, a time-aware LSTM Network addresses predictions on sequential data,

irrespective of the time lag between data points [47]. Finally, an architecture featuring an unidirectional LSTM encoder and a bidirectional LSTM decoder employs an attention mechanism over groups of time sequences, with encoder outputs additively passed to the decoder's input instead of calculating attention over longer sequences [48].

C. Transformer Methods

Transformers have largely supplanted RNN models in sequence modeling tasks due to their proficiency in learning long-range dependencies and interactions in sequential data [49]. However, time series forecasting poses unique challenges for Transformers. Their limitations in effectively modeling complex temporal dependencies and their computational intensity necessitate specialized approaches for this domain. One such approach, LogTrans [50], addresses these issues by employing local convolutions and a log-sparse self-attention mechanism to capture local patterns and reduce space complexity.

Informer [51] is an extension of of the Transformer with sparse self attention to mine the most important temporal features in a sequence. Autoformer [19] uses a decomposition framework and auto-correlation methods from statistical methods to implement an autoregressive attention that exploits the inherent periodicity in a time series data. Building upon Autoformer's decomposition architecture, FEDformer [20] and TDformer [21] enhance time series pattern learning by decoupling trend and seasonality modeling. Both methods leverage Fourier attention in the frequency domain, achieving linear time complexity. However, these approaches do not fully exploit multi-dimensional dependencies, which could further improve long-term forecasting accuracy. To address this, Crossformer [52] utilizes cross-dimensional information for multivariate modeling.

D. Long vs. Short Input Windows and Prediction Horizons

Having reviewed various LSTM and Transformer models in the literature, we turn our attention to a direct comparison of their strengths and weaknesses in sequence modeling. The Transformer-based methods use mainly self-attention and multiple heads to model long-range dependencies with large input windows (e.g., 50-100+). This contrasts with the present study's exploration of LSTM-based encoder-decoder models using shorter input windows (up to 12). Our focus is on their efficacy for localized forecasts: local patterns and shorter prediction horizons where computational efficiency is also a key consideration, reflecting differing typical application contexts.

Why should we pay attention to methods for short-term predictions? Beyond lower computational needs, modeling local patterns is often more critical than global trends for practical reasons. First, many real-world applications, such as hourly traffic management or minute-by-minute financial trading, prioritize accurate immediate forecasts where local patterns are most influential, unlike year-long trend predictions. Secondly, for non-stationary time series whose statistical properties change, global trends can mislead as past behavior may not represent the future; local pattern modeling allows adaptation to the most recent data dynamics. Furthermore, local patterns are crucial for capturing sudden changes, shocks, or anomalies that models focused on global, long-term trends might fail to react to quickly.

E. Evaluation Studies

Often algorithms are evaluated on different datasets or different metrics, or both. Some cases also see the use of these forecasting models on synthetic datasets. Although all forecasting models, citing the shortcomings of each of their predecessors, improve the state-of-the-art results, there is a lack of evaluation on standard real world datasets. The closest relatable work to that of ours is that of [9], who (a) conduct a detailed analysis of various multi-variate time series datasets and their patterns which account for explaining each of the model's performance on that data, (b) presented a prediction analysis over a longer term whilst deploying direct and recursive strategies, and (c) studied the application of Spatio-Temporal Graph Convolutional Networks. Although an evaluative study of non-gated RNNs, LSTMs and GRUs on short term predictions was presented in [53], it did not cover the encoder-decoder or the effect of various attention scoring functions. Our work compares the performances of the existing encoder-decoder models based on LSTM units and the effect of attention scores on short term predictions based on the predictions as a function of varying input sizes. Furthermore, we compare them to Transformer methods to put the results in the context of contemporary research.

III. Methodology

A. Problem Formulation

Consider a series of observed time series vector samples $Y_T = \{y_1, y_2, \cdots, y_T\}$ where $y_t \in \mathbb{R}^n$, n is the number of time series channels. Our objective is to predict a series of H future samples $\tilde{y}_{T+h}$, $h \in \{1, \cdots, H\}$. When the underlying architecture is an encoder-decoder with an RNN such as an LSTM in the decoder, the output is produced in the style of an RNN, viz. one sample at a time. It thus takes H time steps to predict all H samples. We refer to T of the input set Y_T as the *window width* and H of the output set as the *horizon*.

B. LSTM-Based Encoder

We adopt a sequence-to-sequence learning pipeline to encode T historical input variables for a given time window and decode the future h predictions, but with minor modifications. The encoder is an RNN based on LSTM or GRU that encodes the input sequences into a fixed dimensional feature vector [18], [37], [10]. For time series prediction, given the input sequence $Y_T = \{y_1, y_2, \cdots, y_T\}$ with $y_t \in \mathbb{R}^n$, where n is the number of driving variables, the encoder can be applied to learn a mapping from y_t to h_t $h_t = f_1(h_{t-1}; y_t)$ where $h_t \in \mathbb{R}^m$ is the hidden state of the encoder at time t, m is the size of the hidden state, and f_1 is a non-linear

activation function that could be an LSTM or a GRU. In this paper, we use an LSTM unit as f_1 to capture temporal dependencies. Each LSTM unit has a memory cell with the state s_t at time t. Access to the memory cell is controlled by three sigmoid gates: forget gate f_t, input gate i_t and output gate o_t. The update of an LSTM unit can be summarized as follows:

$$f_t = \sigma(W_f[h_{t-1}; y_t] + b_f) \tag{1}$$

$$i_t = \sigma(W_i[h_{t-1}; y_t] + b_i) \tag{2}$$

$$o_t = \sigma(W_o[h_{t-1}; y_t] + b_o) \tag{3}$$

$$s_t = f_t \odot s_{t-1} + i_t \odot \tanh(W_s[h_{t-1}; y_t] + b_s) \tag{4}$$

$$h_t = o_t \odot \tanh(s_t) \tag{5}$$

where $[h_{t-1}; y_t] \in \mathbb{R}^{m+n}$ is a concatenation of the previous hidden state h_{t-1} and the current input y_t. The parameters W_f, W_i, W_o, $W_s \in \mathbb{R}^{m\times(m+n)}$, and b_f, b_i, b_o, $b_s \in \mathbb{R}^m$ are to be learned. The operators σ and $\odot$ are the logistic sigmoid function and the element-wise multiplication, respectively.

C. LSTM-Based Decoder without Attention

To predict the output $\hat{y_T}$, we use another LSTM-based RNN to decode the fixed vector $\mathbb{R}^m$ produced by the encoder LSTM:

$$\hat{y}_T = F(y_{t+1}, y_{t+2}, \cdots, y_{T+h-1}, X_T) \tag{6}$$

where $\hat{y}_T$ is the total number of sequences to be predicted, given the input sequence X_t. We experiment on two such encoder-decoder models without attention mechanisms, viz. that of Cho et al. [18] and Sutskever et al. [10]. The only difference between the two models is that in [18] an extra context vector is augmented to inputs at each of the time steps in the decoder. The context vector being the last hidden state of the encoder. The problem with a vanilla decoder is that its performance can degrade over a longer range of sequences [38].

D. LSTM-Based Decoder with Temporal Attention

The performance of the encoder-decoder network can deteriorate rapidly as the length of the input sequence increases [38]. Therefore, following the encoder with input attention, a temporal attention mechanism is used in the decoder to adaptively select relevant encoder hidden states across all time steps. The attention weight of each encoder hidden state at time t is calculated based upon the previous decoder hidden state $d_{s-1} \in \mathbb{R}^m$ and each of the hidden states of the encoder LSTM h'_t where T is the number of time steps in the input window. The term $[d_{s-1}; h'_t] \in \mathbb{R}^{2m}$ is a concatenation of the previous hidden state of the decoder and hidden states of the LSTM encoder units, whereas $v_d \in \mathbb{R}^m$, $W_d \in \mathbb{R}^{m\times 2m}$ and $U_d \in \mathbb{R}^{m\times m}$ are parameters to learn. The attention schemes per [17], [16] are as follows:

$$\text{score}(d_{s-1}, h'_t) = \begin{cases} v_a^T \sigma(W_d d_{s-1} + U_d h'_t) & \text{Bahdanau} \quad \text{(a)} \\ v_a^T \sigma(W_d[d_{s-1}; h'_t]) & \text{Concat} \quad \text{(b)} \\ d_{s-1}^T W_d h'_t & \text{General} \quad \text{(c)} \\ d_{s-1}^T h'_t & \text{Dot} \quad \text{(d)} \end{cases} \tag{7}$$

The slight changes in eqs. 7a,b are the replacement of the $\tanh$ non-linearity by the sigmoid σ, to keep the range of the intermediary logits in the range $[0, 1]$.

The attention weight β_t^i represents the weight of the ith encoder hidden state for the prediction:

$$\beta_t^s = \frac{\exp(\text{score}(d_{s-1}, h'_t))}{\sum_{k=1}^{T_x} \exp(\text{score}(d_{s-1}, h_t))} \tag{8}$$

Since each encoder hidden state h_i is mapped to a temporal component of the input, the attention mechanism computes the context vector c_t as a weighted sum of all the encoder hidden states $\{h'_1, h'_2, \cdots, h'_T\}$ as

$$c_s = \sum_{t=1}^{T} \beta_t^s h_t. \tag{9}$$

The context vector c_t is distinctly calculated at each time step as per Equation 9.

Once we get the weighted sum of the context vectors, we can combine them with the given target series $\{y_1, y_2, \cdots, y_{T-1}\}$:

$$\tilde{y}_{t-1} = \tilde{w}^T[y_{t-1}; c_{s-1}] + \bar{b} \tag{10}$$

where $[y_{t-1}; c_{t-1}] \in \mathbb{R}^{2m}$ is a concatenation of the decoder input y_{t-1} and the computed context vector c_{t-1}. Parameters $\tilde{w} \in \mathbb{R}^{2m}$ and $\tilde{b} \in \mathbb{R}$ map the concatenation to the size of the decoder input. The newly computed $\tilde{y}_{t-1}$ can be used for the update of the decoder hidden state at time t as :

$$d_s = f_2(d_{s-1}; \tilde{y}_{t-1}) \tag{11}$$

We choose the nonlinear function f_2 as an LSTM unit d_s that can be updated per eqs. 1 to 5.

E. Training Procedure

Our optimization strategy is similar to traditional direct time series forecasting model. Assume the input time series is $Y_t = \{y_1, y_2, \cdots, y_t\}$, we define a tunable window size q, and reformulate the input at time step t as X_t = $\{y_{t-q+1}, y_{t-q+2}, \cdots, y_t\}$. The problem then becomes a regression task with a set of feature-value pairs { X_t , Y_{t+h} } and is optimized with the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = e^{-8}$) [54], a variant of Stochastic Gradient Decent (SGD), with learning rate of e^{-5} to train the model. The minibatch size is set to 32 and number of epochs at 1,000, for training every model across all datasets. The squared error is the default loss function for many forecasting tasks, the corresponding optimization objective is formulated as:

$$\min_{\Theta} \sum_{y \in \Omega_{Train}} \|Y_t - \hat{Y}_{t-h}\|_F^2 \tag{12}$$

where Θ denotes the parameter set of our model, Ω_{Train} is the set of time stamps used for training, $\|\cdot\|_F$ is the Frobenius norm, and h is the horizon. Similar to [10], [18], [17], we train the encoder and decoder models jointly to minimize the objective function.

IV. Evaluation

A. Datasets

The multivariate time series datasets used in our study are as follows: Solar[1], Traffic[2], Electricity[3], Beijing PM2.5[4], and Exchange[5]. The pattern analysis on these datasets helps to understand the performance of time series forecasting models. Temporal regularity has been quantified using the Sample Entropy [9]. Following the methods used in [8] each of the datasets is split into training set (first 60%), validation set (next 20%) and test set (last 20%). All the training, validation and test data were normalized by Min-Max Scaling.

B. Parameter Settings

There are two parameters we have experimented on: the number of time steps for the input window T and the number of time steps to be predicted in the horizon h. Different from refs. [8], [9], [40], we report the results of the experiments where we set the value of $T \in \{3, 6, 9, 12\}$ and $h \in \{3, 6, 9, 12\}$. We use a single layer of LSTM units in both the encoder and decoder LSTMs without any dropout. We have experimented with the number of hidden units in the encoder and the decoder in $\{128, 256, 512\}$. All weights have been initialized from $-\sqrt{1/k}$ to $\sqrt{1/k}$, where k is the number of hidden units. We report average results over two runs of each model.

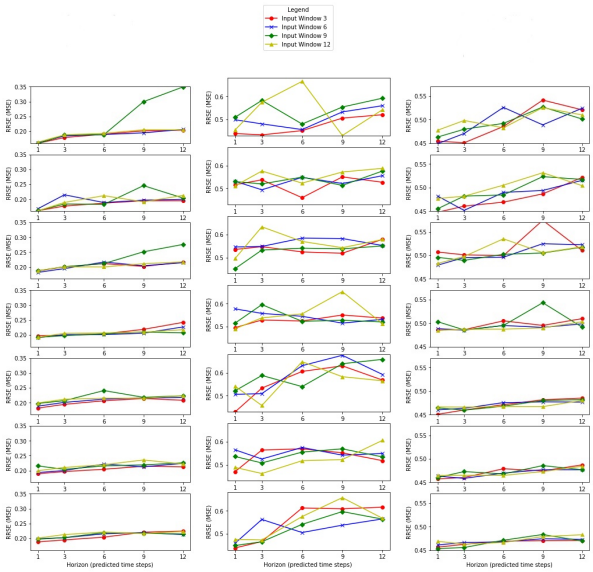
C. Metrics

To keep the evaluation metric standard to the benchmark results reported in [8], [9] we use the metrics in those papers, viz. the Root Relative Squared Error (RSE) and the Empirical Correlation Coefficient (CORR). A model where the RSE exceeds 1 indicates a bad prediction. Lower RSE values implies better predictions, while a higher CORR value is better.

D. Methods for Comparison

The two methods without attention used for comparison in this paper listed in the first two rows of Table I, are labeled as Cho et al. [18], and Sutskever et al. [10]. The two models with attention are Bahdanau et al. [17], and Luong et al. [16]. Of the two with attention, we experimented different attention schemes with corresponding scoring equations, referred to as "Bahdanau et al. (*)" and "Luong et al.(*)" where * refers to the attention scheme in Table I.

[1]https://www.nrel.gov/grid/solar-power-data.html
[2]https://pems.dot.ca.gov/
[3]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[4]https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data
[5]https://github.com/laiguokun/multivariate-time-series-data/tree/master/exchange_rate



(a) Solar (b) Electricity (c) Traffic

Fig. 1: RSE plots (*y-axis*) at different prediction horizon (*x-axis*) for different input window widths (*by color*) evaluated using three datasets (*by column*) of the seven forecasters: Bahdanau et al. using attention schemes "Concat," "General," "Dot"; Cho et al.; Luong et al. using attention schemes "Concat," "General," "Dot." (*top to bottom*).

We compare the results to that of LSTNet [8], selected as our benchmark set. LSTNet serves as a strong, contemporary, and relevant benchmark because it was a leading model in the field at the time, designed to tackle the same core problems, and evaluated using consistent methodologies and datasets.

TABLE I: The methods and the attention scoring functions used in our experiments.

Method Name	Attention Scheme	Scoring Function	Eq.
Cho et al.	N/A	N/A	N/A
Sutskever et al.	N/A	N/A	N/A
Bahdanau et al.	Bahdanau	$v_a^T \sigma(W_d d_{s-1} + U_d h'_t)$	7(a)
Bahdanau et al.	Concat	$v_a^T \sigma(W_d [d_{s-1}; h'_t])$	7(b)
Bahdanau et al.	General	$d_{s-1}^T W_d h'_t$	7(c)
Bahdanau et al.	Dot	$d_{s-1}^T h'_t$	7(d)
Luong et al.	Concat	$v_a^T \sigma(W_d [d_{s-1}; h'_t])$	7(b)
Luong et al.	General	$d_{s-1}^T W_d h'_t$	7(c)
Luong et al.	Dot	$d_{s-1}^T h'_t$	7(d)

E. Discussion and Analysis of Experimental Results

We provide the RSE and CORR values of nine LSTM methods on five datasets in tables II and III. Due to space consideration, we show the results for window size set to 3, 6, 9, and 12 and for horizon of 3, 6, and 12 (skipping 9) for each method. From these tables, we observe that, on the three datasets with high periodicity (viz. Solar, Traffic,

TABLE II: Evaluation summary of all LSTM models (RSE and CORR) trained on MSE (Part I). Results in bold face indicate that the result of that column is better than the benchmark result [8] of a particular metric.

Dataset		Solar			Traffic			Electricity			Exchange-Rate			Beijing PM 2.5		
		Horizon			Horizon			Horizon			Horizon			Horizon		
	Metrics	3	6	12	3	6	12	3	6	12	3	6	12	3	6	12
Method: Cho et al. Next rows correspond to window size of 3, 6, 9, 12																
3	RSE	0.199	**0.203**	**0.242**	0.486	0.505	0.51	0.529	0.524	0.537	1.222	1.154	1.055	0.101	0.077	0.08
	CORR	0.981	**0.981**	**0.973**	**0.879**	0.867	**0.862**	0.838	0.841	0.826	0.604	0.639	0.693	0.944	0.959	0.958
6	RSE	0.201	**0.201**	**0.227**	0.485	0.495	0.499	0.558	0.545	0.532	1.119	1.035	0.916	0.087	0.085	0.091
	CORR	0.982	**0.981**	**0.977**	**0.877**	**0.870**	**0.868**	0.815	0.821	0.834	0.719	0.752	0.464	0.958	0.956	0.957
9	RSE	0.197	**0.203**	**0.207**	0.486	0.495	**0.492**	0.596	0.523	0.52	1.035	0.912	0.813	0.088	0.084	0.1
	CORR	0.982	**0.981**	**0.98**	**0.873**	**0.871**	**0.873**	0.808	0.837	0.829	0.761	0.746	0.788	0.956	0.951	0.955
12	RSE	0.205	**0.206**	**0.217**	0.487	**0.487**	0.503	0.538	0.555	0.513	0.948	0.865	1.158	0.089	0.104	0.119
	CORR	0.981	**0.980**	**0.978**	0.872	**0.874**	**0.866**	0.831	0.825	0.827	0.739	0.747	0.55	0.955	0.956	0.951
Method: Sutskever et al.																
3	RSE	0.208	**0.208**	**0.212**	**0.467**	**0.479**	**0.483**	0.569	0.578	0.539	0.822	1.571	1.552	0.085	0.086	0.736
	CORR	0.980	**0.979**	**0.979**	**0.882**	**0.878**	**0.879**	0.824	0.825	0.839	0.569	0.420	0.196	0.959	0.957	0.955
6	RSE	0.214	**0.211**	0.209	**0.468**	**0.481**	**0.487**	0.567	0.597	0.54	1.410	1.842	1.827	0.777	0.796	0.745
	CORR	0.978	**0.979**	**0.98**	**0.880**	**0.877**	**0.878**	0.823	0.817	0.841	0.368	-0.032	-0.066	0.958	0.953	0.953
9	RSE	0.218	**0.220**	**0.21**	**0.468**	0.475	**0.489**	0.607	0.546	0.577	1.843	1.840	1.813	0.768	0.794	0.736
	CORR	0.978	**0.978**	**0.979**	**0.880**	**0.879**	**0.877**	0.802	0.834	0.819	-0.034	-0.039	-0.058	0.955	0.954	0.957
12	RSE	0.230	**0.219**	**0.214**	0.486	**0.483**	0.496	0.514	0.561	0.574	1.517	1.807	1.835	0.784	0.791	0.789
	CORR	0.974	**0.977**	**0.979**	**0.874**	**0.878**	**0.878**	0.836	0.831	0.822	0.005	-0.017	-0.07	0.954	0.956	0.953
Method: Bahdanau et al. *(Bahdanau)*																
3	RSE	0.182	**0.193**	**0.202**	**0.453**	**0.482**	**0.494**	0.435	0.461	0.499	0.587	0.526	0.507	0.072	0.074	0.078
	CORR	**0.985**	**0.983**	**0.981**	**0.886**	**0.877**	**0.869**	0.866	0.856	0.852	0.872	0.862	0.785	0.959	0.960	0.957
6	RSE	0.184	**0.192**	**0.212**	**0.474**	0.525	0.515	0.497	0.468	0.555	0.598	0.492	0.543	0.089	0.082	0.083
	CORR	**0.985**	**0.983**	**0.979**	**0.875**	0.847	0.857	0.834	0.860	0.842	0.905	0.906	0.833	0.960	0.957	0.957
9	RSE	0.187	**0.192**	**0.21**	0.484	0.493	0.516	0.515	0.510	0.541	0.555	0.339	0.831	0.093	0.089	0.092
	CORR	0.984	**0.983**	**0.981**	0.870	0.869	**0.862**	0.836	0.825	0.836	0.899	0.936	0.83	0.958	0.956	0.959
12	RSE	0.188	**0.202**	**0.203**	0.484	**0.487**	0.51	0.579	0.472	0.553	0.597	0.384	0.467	0.091	0.091	0.1
	CORR	0.984	**0.981**	**0.981**	**0.874**	**0.871**	0.861	0.813	0.850	0.837	0.913	0.903	0.88	0.960	0.958	0.959
Method: Bahdanau et al. *(Concat)*																
3	RSE	**0.179**	**0.190**	**0.204**	**0.451**	**0.486**	0.521	0.433	0.451	0.521	0.690	0.435	0.495	0.075	0.074	0.082
	CORR	**0.985**	**0.983**	**0.981**	**0.886**	**0.874**	0.854	0.873	0.868	0.845	0.868	0.857	0.86	0.959	0.960	0.959
6	RSE	0.185	**0.189**	**0.207**	**0.471**	0.526	0.524	0.481	0.456	0.56	0.550	0.437	0.654	0.085	0.080	0.097
	CORR	0.984	**0.983**	**0.98**	**0.877**	0.850	0.857	0.850	0.856	0.817	0.934	0.859	0.776	0.958	0.958	0.954
9	RSE	0.188	**0.189**	0.349	0.480	0.492	0.501	0.583	0.480	0.593	0.668	0.424	0.539	0.090	0.085	0.102
	CORR	0.984	**0.983**	**0.973**	0.873	0.868	**0.867**	0.802	0.850	0.804	0.903	0.903	0.84	0.958	0.958	0.957
12	RSE	0.188	**0.193**	**0.205**	0.498	**0.483**	0.51	0.575	0.666	0.543	0.568	0.387	0.449	0.099	0.091	0.109
	CORR	0.984	**0.983**	**0.98**	0.867	**0.873**	**0.864**	0.812	0.794	0.827	0.856	0.737	0.824	0.954	0.958	0.957
Method: Bahdanau et al. *(General)*																
3	RSE	**0.177**	**0.187**	**0.195**	**0.460**	**0.469**	0.521	0.538	0.460	0.527	0.630	0.919	0.594	0.078	0.073	0.075
	CORR	**0.985**	**0.984**	**0.983**	**0.883**	**0.880**	0.857	0.834	0.862	0.838	0.896	0.703	0.794	0.959	0.960	0.957
6	RSE	0.215	**0.189**	**0.2**	**0.451**	0.490	0.515	0.494	0.548	0.556	0.587	0.409	0.477	0.087	0.087	0.113
	CORR	0.983	**0.984**	**0.982**	**0.887**	**0.870**	0.857	0.837	0.824	0.815	0.904	0.873	0.858	0.959	0.959	0.949
9	RSE	0.184	**0.183**	**0.204**	0.482	**0.484**	0.517	0.520	0.549	0.576	0.502	0.442	0.631	0.096	0.086	0.094
	CORR	0.985	**0.984**	**0.98**	0.872	**0.870**	**0.862**	0.843	0.822	0.807	0.903	0.906	0.852	0.959	0.960	0.958
12	RSE	0.189	**0.212**	**0.213**	0.482	0.505	0.504	0.576	0.523	0.588	0.575	0.647	0.514	0.098	0.103	0.094
	CORR	0.984	**0.980**	**0.979**	0.872	0.867	**0.862**	0.818	0.836	0.808	0.893	0.780	0.786	0.960	0.954	0.957
Method: Bahdanau et al. *(Dot)*																
3	RSE	0.200	**0.212**	**0.216**	0.501	0.499	0.51	0.547	0.525	0.58	1.298	0.656	1.119	0.080	0.076	0.088
	CORR	0.981	**0.981**	**0.979**	**0.874**	**0.870**	0.861	0.829	0.840	0.814	0.496	0.736	0.444	0.956	0.958	0.957
6	RSE	0.195	**0.217**	**0.215**	0.495	0.496	0.522	0.548	0.585	0.552	0.809	0.970	0.933	0.085	0.085	0.087
	CORR	0.982	**0.978**	**0.979**	**0.876**	0.869	0.859	0.826	0.806	0.828	0.812	0.582	0.616	0.958	0.955	0.953
9	RSE	0.201	**0.212**	**0.276**	0.489	0.502	0.517	0.533	0.541	0.551	0.594	0.809	0.698	0.095	0.087	0.117
	CORR	0.982	**0.981**	**0.97**	**0.875**	0.868	0.859	0.829	0.833	0.836	0.745	0.732	0.5	0.953	0.960	0.958
12	RSE	0.201	**0.200**	**0.217**	0.498	0.535	0.518	0.634	0.570	0.579	0.691	0.682	0.876	0.094	0.105	0.097
	CORR	0.981	**0.981**	**0.979**	**0.873**	0.849	0.859	0.803	0.806	0.818	0.855	0.839	0.776	0.954	0.957	0.954

TABLE III: Evaluation summary of all models (RSE and CORR) trained on MSE (Part II). Results in bold face indicate that the result of that column is better than the benchmark result [8] of a particular metric

Dataset		Solar			Traffic			Electricity			Exchange-Rate			Beijing PM 2.5		
		Horizon			Horizon			Horizon			Horizon			Horizon		
	Metrics	3	6	12	3	6	12	3	6	12	3	6	12	3	6	12
Method: Luong et al. *(Concat)*																
3	RSE	0.194	**0.206**	**0.208**	**0.460**	**0.471**	**0.485**	0.533	0.605	0.568	1.647	0.876	1.674	0.085	0.080	0.081
	CORR	0.982	**0.980**	**0.979**	**0.884**	**0.879**	**0.876**	0.821	0.807	0.821	0.743	0.501	-0.031	0.956	0.960	0.956
6	RSE	0.201	**0.212**	**0.217**	**0.463**	**0.475**	**0.477**	0.509	0.630	0.592	1.117	1.053	1.225	0.090	0.095	0.087
	CORR	0.981	**0.979**	**0.977**	**0.884**	**0.878**	**0.879**	0.830	0.812	0.815	0.696	0.630	0.617	0.956	0.959	0.959
9	RSE	0.207	**0.241**	**0.223**	**0.459**	**0.468**	**0.482**	0.588	0.539	0.658	0.893	0.898	0.971	0.097	0.095	0.091
	CORR	0.980	**0.976**	**0.977**	**0.882**	**0.880**	**0.876**	0.814	0.832	0.786	0.697	0.671	0.684	0.961	0.959	0.954
12	RSE	0.212	**0.215**	**0.223**	**0.466**	**0.467**	**0.48**	0.457	0.648	0.565	0.683	1.437	0.82	0.100	0.108	0.226
	CORR	0.979	**0.978**	**0.977**	**0.881**	**0.882**	**0.877**	0.852	0.800	0.818	0.724	-0.174	0.692	0.956	0.956	0.786
Method: Luong et al. *(General)*																
3	RSE	0.197	**0.204**	**0.212**	**0.460**	**0.479**	**0.487**	0.564	0.569	0.518	0.815	0.823	1.04	0.089	0.084	0.088
	CORR	0.982	**0.980**	**0.978**	**0.884**	**0.880**	**0.876**	0.825	0.812	0.842	0.862	0.609	0.637	0.959	0.959	0.958
6	RSE	0.200	**0.222**	**0.224**	**0.459**	**0.470**	**0.477**	0.525	0.575	0.549	0.883	0.909	1.044	0.092	0.111	0.083
	CORR	0.981	**0.978**	**0.977**	**0.884**	**0.879**	**0.878**	0.831	0.831	0.829	0.764	0.623	0.715	0.955	0.955	0.957
9	RSE	0.204	**0.216**	**0.226**	**0.473**	**0.468**	**0.477**	0.508	0.555	0.534	0.982	1.191	1.604	0.101	0.093	0.104
	CORR	0.980	**0.978**	**0.976**	**0.878**	**0.880**	**0.878**	0.843	0.818	0.84	0.741	0.652	0.649	0.945	0.952	0.953
12	RSE	0.210	**0.219**	**0.222**	**0.465**	**0.465**	**0.484**	0.462	0.517	0.606	1.123	0.927	1.625	0.123	0.106	0.4
	CORR	0.979	**0.978**	**0.977**	**0.881**	**0.882**	**0.875**	0.856	0.842	0.818	0.695	0.604	-0.11	0.952	0.956	0.767
Method: Luong et al. *(Dot)*																
3	RSE	0.195	**0.204**	**0.224**	**0.463**	**0.469**	**0.471**	0.466	0.612	0.616	0.853	0.805	0.65	0.084	0.111	0.784
	CORR	0.982	**0.980**	**0.977**	**0.883**	**0.881**	**0.88**	0.860	0.809	0.807	0.575	0.686	0.763	0.958	0.950	0.016
6	RSE	0.202	**0.215**	**0.213**	**0.466**	**0.468**	**0.473**	0.562	0.505	0.565	0.604	0.861	0.855	0.091	0.090	0.111
	CORR	0.981	**0.978**	**0.979**	**0.880**	**0.880**	**0.882**	0.832	0.848	0.827	0.782	0.667	0.651	0.954	0.958	0.953
9	RSE	0.203	**0.219**	**0.215**	**0.456**	**0.471**	**0.47**	0.465	0.541	0.563	0.934	0.854	1.018	0.099	0.106	0.782
	CORR	0.981	**0.977**	**0.978**	**0.886**	**0.881**	**0.881**	0.858	0.830	0.82	0.596	0.689	0.482	0.958	0.954	0.097
12	RSE	0.213	**0.221**	**0.223**	**0.463**	**0.467**	**0.483**	0.473	0.575	0.567	1.159	0.984	0.869	0.109	0.117	0.166
	CORR	0.979	**0.978**	**0.978**	**0.882**	**0.881**	**0.879**	0.852	0.824	0.824	0.765	0.566	0.749	0.955	0.958	0.809

Beijing PM2.5 datasets), the NMT models used for forecasting outperform the benchmark results as reported in [8]. Experimenting with the hidden units in the encoder and decoder LSTMs did not yield a significant variation in MSE results. The number of hidden units of both the encoder and decoder LSTMs is fixed at 512, to give the models an increased learning capacity.

In Fig. 1, we note that the RSE values across the horizons show rising, though insignificantly so, trends. The Electricity data set results show the most fluctuations. Interestingly, the method without an attention mechanism (Cho et al.; fourth plots from the top) show no substantial deterioration with longer horizon, comparable to Luong et al.'s methods. Comparing Bahdanau et al. and Luong et al., across the data sets and across the attention scoring function, Luong et al. is more stable relative to horizon and window widths.

We note that Bahdanau's model, [17] coupled with the attention scoring scheme (b) as shown in tables II and III performs the best across all the datasets among all models. The Sutskever et al. model [10] has the worst performance across all the datasets. We hypothesize this happens mainly because in this model the decoder is not augmented with any context vector as opposed to the other NMT models. Although Cho's model [18] lacks an attention mechanism, the decoder is augmented with the hidden state of the encoder's last time step. This suggests that augmenting the decoder with a context is essential for better performance in NMT models.

We observe that given a short input to NMT models for predicting a longer horizon, for example, given an input of 3 time steps we need to predict 12 time steps, yields the best results. To date, there exists no work which has experimented with LSTM models for forecasting a longer horizon given a shorter window. Majority of forecasting models have a longer input sequence of {24, 36, 64, 128} time steps, to predict a future horizon of {12, 24, 48} time steps. Our experiments show that NMT models can not only improve on the benchmark method [8] but also handle these cases with decent results in other datasets with no periodic trends. Another question pointed out in Section I is how each model behaves if a different attention scoring function is used? It was mentioned earlier that Bahdanau's model performs best with the "concat (b)" function in Eq. 7. Hence, the choice of attention function is a factor in improving the prediction accuracy. All these observations make us believe that fine tuning of hyper-parameters, employing regularization techniques of the NMT models could lead to improved results across the datasets.

F. Comparison with Transformer Models

We selected three Transformer models, viz. Autoformer [19], FEDformer [20], and TDformer [21], for comparison. They are representative of the decomposition-centric transformer models. They were configured and evaluated specifically to operate within the same short input window widths and prediction horizons as the LSTM models. We

emphasize that this is not testing these Transformers models at their optimal, long-sequence capacity but rather assessing their performance and efficiency in the localized forecasting context where LSTMs are hypothesized to be strong. Of the models being evaluated, FEDformer has two variants in the Fourier and wavelet domains. TDformer can operate in the time, Fourier, and wavelet domains. The results of evaluating these 6 transformer models using the Traffic and Electricity datasets are shown in Table IV. The Transformer studies as reported [19], [20], [21] have a different preprocessing protocol compared to the earlier LSTM methods. In particular, they do not use min-max scaling. Instead, all data—even before being split into training, validation, and test sets—are normalized by the overall maximum value of the series. To ensure a fair comparison, we repeated the experiments for the LSTM with attention methods using the Transformer studies pre-processing pipeline. The results are shown in Table V.

Comparing tables IV and V, we can see that TDformer has the worst performance among all evaluated forecasters. The LSTM methods are competitive (Electricity data set) or slightly better (Traffic) compared to the Autoformer and FEDformer methods. The Bahdanau methods (all four scoring functions) performed the best among the evaluated forecasters.

G. Computational Cost

The two models with attention mechanisms [17], [16] generally require more computational cost than the two without attention mechanisms. Attention mechanisms introduce additional computations to calculate attention weights and apply them to the encoder hidden states. These additional steps increase the overall complexity of the model compared to those without attention, which primarily consist of LSTM operations. The computational cost of the attention mechanism scales with the sequence length, as the model needs to compute attention weights for each decoder step over all encoder hidden states.

The computational cost of simpler attention scoring functions tend to be less computationally expensive. The "dot (d)" product attention in Eq.7 is generally the simplest and fastest, as it only involves matrix multiplications. The "Concat (b)," "General (c)," and "Bahdanau (a)" attention in Eq. 8 involve additional linear transformations (W_d, U_d) and may be slightly more expensive. Other attention mechanisms that involve more complex calculations will increase the computational cost. While there might be some differences in computational cost between different attention scoring functions, the overall impact on the model's computational cost is likely to be less significant than other factors, such as the LSTM layer sizes or the input sequence length.

When comparing the computational cost of LSTM-based models with attention mechanisms to a Transformer-based model, even with the same input window width and output prediction horizon, there are some key differences. The first is in the attention complexity. LSTM attention typically calculates attention between the decoder's current state and

TABLE IV: Evaluation summary of Transformer Models (RSE and CORR) using Transformer studies preprocessing pipeline and trained on MSE and trained on MSE.

Dataset		Traffic			Electricity		
		Horizon			Horizon		
	Metrics	3	6	12	3	6	12
Method: Autoformer							
3	RSE	0.706	0.807	0.89	0.419	0.535	0.761
	CORR	0.793	0.722	0.654	0.889	0.827	0.684
6	RSE	0.732	0.775	0.83	0.496	0.543	0.595
	CORR	0.771	0.742	0.694	0.849	0.822	0.791
9	RSE	0.685	0.742	0.78	0.414	0.431	0.516
	CORR	0.81	0.772	0.743	0.895	0.888	0.846
12	RSE	0.672	0.711	0.733	0.442	0.428	0.474
	CORR	0.825	0.798	0.779	0.883	0.89	0.867
Method: FEDformer- Fourier							
3	RSE	0.659	0.758	0.91	0.386	0.464	0.579
	CORR	0.832	0.773	0.674	0.905	0.869	0.809
6	RSE	0.667	0.733	0.798	0.385	0.44	0.507
	CORR	0.832	0.79	0.758	0.907	0.882	0.848
9	RSE	0.666	0.707	0.734	0.391	0.428	0.471
	CORR	0.833	0.813	0.796	0.905	0.888	0.867
12	RSE	0.663	0.691	0.703	0.394	0.424	0.446
	CORR	0.836	0.821	0.81	0.904	0.89	0.88
Method: FEDformer- wavelet							
3	RSE	0.675	0.79	0.905	0.396	0.5	0.645
	CORR	0.824	0.74	0.63	0.9	0.848	0.753
6	RSE	0.687	0.771	0.826	0.393	0.464	0.558
	CORR	0.82	0.757	0.706	0.902	0.868	0.814
9	RSE	0.682	0.736	0.729	0.4	0.452	0.457
	CORR	0.82	0.784	0.801	0.9	0.875	0.875
12	RSE	0.684	0.712	0.699	0.403	0.441	0.437
	CORR	0.822	0.803	0.814	0.899	0.881	0.884
Method: TDformer- Fourier							
3	RSE	0.657	0.885	1.118	0.657	0.885	1.118
	CORR	0.737	0.527	0.224	0.737	0.527	0.224
6	RSE	0.731	0.888	1.04	0.731	0.888	1.04
	CORR	0.664	0.484	0.264	0.664	0.484	0.264
9	RSE	0.777	0.881	0.924	0.777	0.881	0.924
	CORR	0.596	0.445	0.375	0.596	0.445	0.375
12	RSE	0.778	0.845	0.801	0.778	0.845	0.801
	CORR	0.58	0.466	0.54	0.58	0.466	0.54
Method: TDformer- wavelet							
3	RSE	0.662	0.897	1.149	0.662	0.897	1.149
	CORR	0.733	0.513	0.182	0.733	0.513	0.182
6	RSE	0.733	0.907	1.069	0.733	0.907	1.069
	CORR	0.663	0.458	0.212	0.663	0.458	0.212
9	RSE	0.778	0.884	0.941	0.778	0.884	0.941
	CORR	0.595	0.443	0.344	0.595	0.443	0.344
12	RSE	0.77	0.84	0.807	0.77	0.84	0.807
	CORR	0.588	0.475	0.53	0.588	0.475	0.53
Method: TDformer- time							
3	RSE	0.658	0.886	1.115	0.658	0.886	1.115
	CORR	0.736	0.526	0.228	0.736	0.526	0.228
6	RSE	0.727	0.897	1.035	0.727	0.897	1.035
	CORR	0.671	0.467	0.271	0.671	0.467	0.271
9	RSE	0.769	0.899	0.938	0.769	0.899	0.938
	CORR	0.609	0.417	0.349	0.609	0.417	0.349
12	RSE	0.768	0.834	0.831	0.768	0.834	0.831
	CORR	0.591	0.485	0.489	0.591	0.485	0.489

TABLE V: Evaluation summary of LSTM Models (RSE and CORR) using Transformer studies preprocessing pipeline and trained on MSE.

Dataset		Traffic			Electricity		
		Horizon			Horizon		
	Metrics	3	6	12	3	6	12
Method: Bahdanau et al. (Bahdanau)							
3	RSE	0.599	0.6	0.596	0.368	0.373	0.372
	CORR	0.869	0.87	0.872	0.928	0.927	0.927
6	RSE	0.628	0.634	0.636	0.406	0.415	0.418
	CORR	0.854	0.851	0.849	0.914	0.91	0.909
9	RSE	0.644	0.659	0.669	0.427	0.443	0.448
	CORR	0.848	0.838	0.831	0.906	0.898	0.896
12	RSE	0.647	0.659	0.665	0.432	0.449	0.456
	CORR	0.847	0.837	0.833	0.904	0.895	0.894
Method: Bahdanau et al. (Concat)							
3	RSE	0.6	0.599	0.598	0.369	0.372	0.373
	CORR	0.868	0.87	0.872	0.927	0.927	0.927
6	RSE	0.628	0.636	0.638	0.405	0.415	0.419
	CORR	0.854	0.849	0.847	0.914	0.909	0.909
9	RSE	0.645	0.658	0.671	0.428	0.442	0.447
	CORR	0.847	0.838	0.829	0.905	0.898	0.896
12	RSE	0.645	0.664	0.664	0.432	0.452	0.459
	CORR	0.645	0.836	0.832	0.904	0.895	0.892
Method: Bahdanau et al. (General)							
3	RSE	0.596	0.598	0.595	0.369	0.372	0.372
	CORR	0.869	0.871	0.872	0.928	0.926	0.927
6	RSE	0.628	0.635	0.635	0.407	0.417	0.414
	CORR	0.853	0.849	0.848	0.914	0.909	0.91
9	RSE	0.644	0.659	0.668	0.428	0.444	0.445
	CORR	0.847	0.838	0.831	0.904	0.897	0.897
12	RSE	0.646	0.661	0.67	0.435	0.451	0.455
	CORR	0.846	0.837	0.831	0.903	0.894	0.894
Method: Bahdanau et al. (Dot)							
3	RSE	0.597	0.599	0.596	0.37	0.373	0.373
	CORR	0.869	0.869	0.872	0.928	0.926	0.927
6	RSE	0.628	0.638	0.639	0.405	0.417	0.416
	CORR	0.852	0.847	0.845	0.914	0.909	0.91
9	RSE	0.648	0.662	0.668	0.43	0.445	0.447
	CORR	0.846	0.835	0.831	0.904	0.897	0.896
12	RSE	0.651	0.666	0.674	0.436	0.454	0.456
	CORR	0.844	0.834	0.828	0.901	0.893	0.894
Method: Luong et al. (Concat)							
3	RSE	0.648	0.646	0.644	0.498	0.498	0.496
	CORR	0.855	0.856	0.857	0.878	0.878	0.88
6	RSE	0.65	0.654	0.652	0.505	0.511	0.509
	CORR	0.854	0.85	0.851	0.875	0.871	0.872
9	RSE	0.65	0.655	0.663	0.504	0.513	0.516
	CORR	0.854	0.85	0.846	0.875	0.87	0.869
12	RSE	0.65	0.653	0.659	0.504	0.513	0.519
	CORR	0.854	0.851	0.848	0.875	0.869	0.868
Method: Luong et al. (General)							
3	RSE	0.65	0.647	0.644	0.5	0.5	0.497
	CORR	0.854	0.855	0.857	0.879	0.878	0.878
6	RSE	0.651	0.654	0.654	0.504	0.509	0.509
	CORR	0.853	0.851	0.85	0.875	0.872	0.872
9	RSE	0.651	0.657	0.663	0.505	0.513	0.517
	CORR	0.853	0.849	0.844	0.875	0.87	0.867
12	RSE	0.652	0.656	0.661	0.507	0.515	0.519
	CORR	0.853	0.849	0.844	0.874	0.869	0.868
Method: Luong et al. (Dot)							
3	RSE	0.649	0.647	0.646	0.5	0.499	0.496
	CORR	0.854	0.856	0.857	0.877	0.878	0.878
6	RSE	0.651	0.654	0.655	0.505	0.512	0.511
	CORR	0.852	0.85	0.85	0.874	0.87	0.871
9	RSE	0.652	0.659	0.666	0.506	0.517	0.518
	CORR	0.852	0.847	0.841	0.875	0.867	0.866
12	RSE	0.651	0.657	0.665	0.506	0.518	0.52
	CORR	0.852	0.848	0.842	0.874	0.867	0.866

all encoder hidden states. Transformers use self-attention, where each position in the input sequence attends to all other positions. This self-attention in Transformers has a quadratic complexity ($O(T^2)$) with respect to the input window width T.

Also worth noting is that LSTMs are inherently sequential in that calculations at each time step depend on the previous time step, thus limiting parallelization. Transformers can process all positions in the input sequence in parallel, which can be much faster, especially with hardware acceleration such as GPUs, but this parallelization comes at the cost of increased memory and computations.

For shorter input window widths, the computational cost might be comparable, with LSTMs slightly cheaper. As the input window width increases, the quadratic complexity of Transformer self-attention becomes dominant, making Transformers more computationally expensive.

V. Conclusion

Our empirical evaluation confirms that LSTM-based encoder-decoder networks can be a powerful tool for time series forecasting, achieving competitive results with careful selection of input window size and attention mechanisms. The key takeaway is that the choice between LSTM-based models and Transformer-based methods is highly context-dependent.

When computational resources are limited or the forecasting task primarily relies on short-term dependencies, LSTM-based models provide an efficient and effective solution. Conversely, for applications demanding the modeling of intricate long-range relationships and the generation of long-horizon predictions, Transformer-based architectures are likely more appropriate. Further investigation is needed to establish clear guidelines for selecting the optimal model based on the specific characteristics of the time series data and the forecasting objectives.

There are areas for future work that would extend our findings. A more extensive set of experiments, utilizing a larger number of random starting points for each model configuration, will enable a robust statistical analysis, including the calculation of confidence intervals and the application of significance tests (e.g., t-tests). This approach will more formally validate the performance differences between LSTM and Transformer-based models. Furthermore, while we provided a theoretical discussion of complexity to advocate for LSTM's computational efficiency, a direct quantitative comparison would offer more definitive evidence. This analysis might involve measuring and reporting metrics such as training and inference latency and memory consumption for all tested models under the same hardware and software conditions.

Acknowledgments

The authors thank the anonymous reviewers for their insightful feedback and constructive suggestions. This work is supported by the U.S. National Science Foundation under

grant number OIA-1946231 and the Louisiana Board of Regents for the Louisiana Materials Design Alliance (LAMDA).

REFERENCES

[1] Y. C. Fung and B. Amonov, "Decoding financial data: Machine learning approach to predict trading actions," in *Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS)*, ser. Annals of Computer Science and Information Systems, M. Bolanowski, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 39. IEEE, 2024, p. 739–744. [Online]. Available: http://dx.doi.org/10.15439/2024F4556

[2] C. Lin, "Key financial indicators analysis and stock trend forecasting based on a wrapper feature selection method," in *Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS)*, ser. Annals of Computer Science and Information Systems, M. Bolanowski, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 39. IEEE, 2024, p. 755–759. [Online]. Available: http://dx.doi.org/10.15439/2024F3560

[3] V.-T. Duong, D.-T.-A. Nguyen, T.-T.-H. Pham, V.-H. Nguyen, and V.-Q. A. Le, "Comparative study of deep learning models for predicting stock prices," in *Proceedings of the Seventh International Conference on Research in Intelligent and Computing in Engineering*, ser. Annals of Computer Science and Information Systems, V. K. Solanki and B. T. Thanh, Eds., vol. 33. PTI, 2022, p. 103–108. [Online]. Available: http://dx.doi.org/10.15439/2022R02

[4] J. Jenko, J. P. Costa, D. Vladušič, U. Bavčar, and R. Šabarkapa, "Learning from the COVID-19 pandemic to improve critical infrastructure resilience using temporal fusion transformers," in *Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS)*, 2024, pp. 375–384.

[5] S. Yang, M. Tan, S. Xia, and F. Liu, "A method of intrusion detection based on attention-lstm neural network," in *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, ser. ICMLT 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 46–50. [Online]. Available: https://doi.org/10.1145/3409073.3409096

[6] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017, online Real-Time Learning Strategies for Data Streams. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217309864

[7] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," *Computing Research Repository (CoRR)*, vol. abs/1607.00148, 2016. [Online]. Available: http://arxiv.org/abs/1607.00148

[8] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 95–104. [Online]. Available: https://doi.org/10.1145/3209978.3210006

[9] J. Yin, W. Rao, M. Yuan, J. Zeng, K. Zhao, C. Zhang, J. Li, and Q. Zhao, "Experimental study of multivariate time series forecasting models," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2833–2839. [Online]. Available: https://doi.org/10.1145/3357384.3357826

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

[11] S. Afreen, N. T. D. Linh, S. Kodur, and A. Begum, "Seq2Seq transformer-based model for optimized Chinese-to-English translation," in *Proceedings of the Ninth International Conference on Research in Intelligent Computing in Engineering*, ser. Annals of Computer Science and Information Systems, V. K. Solanki, T. D. Tan, P. Kumar, and M. Cardona, Eds., vol. 42. PTI, 2024, p. 1–10. [Online]. Available: http://dx.doi.org/10.15439/2024R81

[12] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[13] T. Markovic, A. Dehlaghi-Ghadim, M. Leon, A. Balador, and S. Punnekkat, "Time-series anomaly detection and classification with long short-term memory network on industrial manufacturing systems," in *Proceedings of the 18th Conference on Computer Science and Intelligence Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 35. IEEE, 2023, p. 171–181. [Online]. Available: http://dx.doi.org/10.15439/2023F5263

[14] P. Lam, L. Pham, T. Nguyen, H. Tang, M. Seidl, M. Andresel, and A. Schindler, "LSTM-based deep neural network with a focus on sentence representation for sequential sentence classification in medical scientific abstracts," in *Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS)*, ser. Annals of Computer Science and Information Systems, M. Bolanowski, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 39. IEEE, 2024, p. 219–224. [Online]. Available: http://dx.doi.org/10.15439/2024F5872

[15] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Neural Information Processing Systems*, 2017.

[16] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421. [Online]. Available: https://www.aclweb.org/anthology/D15-1166

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, Jan. 2015, 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

[18] 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 *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[19] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Neural Information Processing Systems*, 2021.

[20] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research. PMLR, 2022.

[21] X. Zhang, X. Jin, K. Gopalswamy, G. Gupta, Y. Park, X. Shi, H. Wang, D. C. Maddix, and Y. Wang, "First de-trend then attend: Rethinking attention for time-series forecasting," *arXiv preprint arXiv:2212.08151*, 2022.

[22] G. E. P. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American Statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1970.10481180

[23] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 3rd ed. USA: Prentice Hall PTR, 1994.

[24] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 847–855.

[25] P. R. Winters, *Forecasting Sales by Exponentially Weighted Moving Averages*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1976, pp. 384–386. [Online]. Available: https://doi.org/10.1007/978-3-642-51565-1_116

[26] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207003001134

[27] I. Melnyk and A. Banerjee, "Estimating structured vector autoregressive models," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 830–839. [Online]. Available: http://proceedings.mlr.press/v48/melnyk16.html

[28] H. Qiu, S. Xu, F. Han, H. Liu, and B. Caffo, "Robust estimation of transition matrices in high dimensional heavy-tailed vector autoregressive processes," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1843–1851. [Online]. Available: http://proceedings.mlr.press/v37/qiu15.html

[29] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian inference and learning in Gaussian process state-space models with particle MCMC," *arXiv:1306.2861 [stat.ML]*, 2013.

[30] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/036402139090002E

[31] J. Connor, L. Atlas, and D. Martin, "Recurrent networks and narma modeling," in *Advances in Neural Information Processing Systems*, J. Moody, S. Hanson, and R. P. Lippmann, Eds., vol. 4. Morgan-Kaufmann, 1992. [Online]. Available: https://proceedings.neurips.cc/paper/1991/file/5ef0b4eba35ab2d6180b0bca7e46b6f9-Paper.pdf

[32] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, "Learning longer memory in recurrent neural networks," in *Workshop at the International Conference on Learning Representation; ICLR 2015*, 12 2015.

[33] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Trans. Neur. Netw.*, vol. 5, no. 2, p. 157–166, Mar. 1994. [Online]. Available: https://doi.org/10.1109/72.279181

[34] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, p. 107–116, Apr. 1998. [Online]. Available: https://doi.org/10.1142/S0218488598000094

[35] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 843–852.

[36] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in lstms for activity detection and early detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1942–1950.

[37] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 3104–3112.

[38] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111. [Online]. Available: https://www.aclweb.org/anthology/W14-4012

[39] O. Hamel and M. Fareh, "Encoder-decoder neural network with attention mechanism for types detection in linked data," in *Proceedings of the 17th Conference on Computer Science and Intelligence Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 30. IEEE, 2022, p. 733–739. [Online]. Available: http://dx.doi.org/10.15439/2022F209

[40] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, p. 2627–2633.

[41] S. Huang, D. Wang, X. Wu, and A. Tang, "DSANet: Dual self-attention network for multivariate time series forecasting," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2129–2132. [Online]. Available: https://doi.org/10.1145/3357384.3358132

[42] T. Lin, T. Guo, and K. Aberer, "Hybrid neural networks for learning the trend in time series," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2273–2279. [Online]. Available: https://doi.org/10.24963/ijcai.2017/316

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[44] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207019301888

[45] Y. G. Cinar, H. Mirisaee, P. Goswami, E. Gaussier, A. Aït-Bachir, and V. Strijov, "Position-based content attention for time series forecasting with sequence-to-sequence rnns," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham: Springer International Publishing, 2017, pp. 533–544.

[46] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv: Machine Learning*, 2017.

[47] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, "Patient subtyping via time-aware LSTM networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 65–74. [Online]. Available: https://doi.org/10.1145/3097983.3097997

[48] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang, "Multi-horizon time series forecasting with temporal attention learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2527–2535. [Online]. Available: https://doi.org/10.1145/3292500.3330662

[49] K. Kaczmarek, J. Pokrywka, and F. Graliński, "Using transformer models for gender attribution in polish," in *Proceedings of the 17th Conference on Computer Science and Intelligence Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, Eds., vol. 30. IEEE, 2022, p. 73–77. [Online]. Available: http://dx.doi.org/10.15439/2022F197

[50] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf

[51] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *AAAI Conference on Artificial Intelligence*, 2020.

[52] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *International Conference on Learning Representation*, 2023.

[53] F. M. Bianchi, E. Maiorino, M. Kampffmeyer, A. Rizzi, and R. Jenssen, *Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis*. Springer, 01 2017.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." *Computing Research Repository (CoRR)*, vol. abs/1412.6980, 2014. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1412.html#KingmaB14