Auto-Lag Networks for Real Valued Sequence to Sequence Prediction

Chapter · September 2019

DOI: 10.1007/978-3-030-30490-4_33

2 authors, including:

Gilles Madi Wamba
Parashift.io
7 PUBLICATIONS **66** CITATIONS

SEE PROFILE

Auto-Lag Networks for Real Valued Sequence to Sequence Prediction

Gilles Madi Wamba* and Nicolas Gaude*

Prevision.io, Paris, France.
{gilles.madi,nicolas.gaude}@prevision.io

Abstract. Many machine learning problems involve predicting a sequence of future values of a target variable. State-of-the-art approaches for such use cases involve LSTM based sequence to sequence models. To improve they performances, those models generally use lagged values of the target variable as additional input features. Therefore, appropriate lag factor has to be chosen during feature engineering. This choice often requires business knowledge of the data. Furthermore, state-of-the-art sequence to sequence models are not designed to naturally handle hierarchical time series use cases. In this paper, we propose a novel architecture that naturally handles hierarchical time series. The contribution of this paper is thus two-folds. First we show the limitations of classical sequence to sequence models in the case of problems involving a real valued target variable, namely the error accumulation problem and we propose a novel LSTM based approach to overcome those limitations. Second, we highlight the limitations of manually selecting fixed lag values to improve the performance of a model. We then use an attention mechanism to introduce a dynamic and automatic lag factor selection that overcomes the former limitations, and requires no business knowledge of the data. We call this architecture Auto-Lag Network (AL-Net). We finally validate our Auto-Lag Net model against state-of-the-art results.

Keywords: Auto-Lag nets · Time series · LSTM · Sequence to Sequence

1 Introduction and Related Work

Over the recent years problems involving time series [1], or sequence to sequence prediction have been of great interest. Several LSTM based approaches have been proposed in the case of text translation [2] [3] and image captioning [4]. Those models rely on a discrete representation of the words using embeddings. In the case of real valued sequence prediction, the values are not discrete. We refer to those problems as real valued sequence to sequence prediction problems.

1.1 Real Valued Time Series Forecasting

Time series forecasting involves observing values of a given measure over a period of time and predicting future values. They are well suited to model data such as weather, stock prices, retail sales or resources consumption [5].

A major application of real valued sequence to sequence prediction is the unit commitment problem. Unit commitment problem refers to a sets of problems where an

energy supplier has to calibrate its production with respects to a set of constraints [6]. In this context, Bouktif et al. [7] propose an LSTM based real valued sequence to sequence prediction model. Their state-of-the-art model uses an orginal genetic algorithm for lag selection to forecast the next 30 minutes time step energy consumption value. However, in order to align the production of the energy with the demand, it is unrealistic to forecast only one 30 minutes time step ahead. In order to remain in realistic conditions, we evaluate our AL-Net model on the forecast of 48 time steps of 30 minute each i.e we use our model to forecast a whole day of energy consumption at a granularity of 30 minutes. Although the promising results on this configuration, we also evaluate the AL-Net model against the state-of-the-art model in [7], for the case where one needs only one time step ahead forecast. Experimental results on a real-world data set shows that our AL-Net model outperforms theirs by up to 15%.

In some situations, one may wants to group several time series following some predefined criteria, we refer to them as grouped or hierarchical time series.

1.2 Hierarchical Time Series Forecasting

Hierarchical time series is a term that designates a set of time series that are hierarchically organized. The hierarchy is generally specified by geographical criteria or by product types (groups). There exists 3 main approaches that can be used to forecast hierarchical time series [8].

- **Bottom-up approach:** It consists in building several models to make low level predictions that can be aggregated into high level predictions. The need to build several models for each group is resource consuming as the number of group might be huge, and leads to error accumulation.
- **Top-down approach:** Here a single model makes high level predictions that are further disaggregated using another model like regression. The major issue is the loss of information relative to individual series dynamics [9].
- **Middle-out approach:** It is an hybridation of the Bottom-up and the Top-down approaches.

With AL-Net, we propose a novel **one-shot** approach. In the one-shot approach, a single model is built and is able to make prediction at each level of the time series hierarchy. In order to achieve this, non temporal data, i.e features specifying the group of each time series are used both at train and prediction time. We thus avoid all the issues inherent to Bottom-up and Top-down aproaches : no error is accumulated, no extra-resource is used to build and train several models, and finally, information relative to individual time series dynamics is preserved and taken into consideration. The evaluation of our one-shot approach on a real hierarchical time series use case [10] shows promising results.

The contribution of this paper is thus two-folds. First we show the limitations of classical sequence to sequence models in the case of problems involving a real valued target variable, and we propose a novel approach to overcome them. Second, we highlight the limitations of selecting fixed lag factors to improve the performance of a model. We then use an attention mechanism to introduce a dynamic lag factor selection that overcome the former limitations, and requires no business knowledge of the

data. The proposed model also introduces a novel one shot-approach for hierarchical or grouped time series prediction. The rest of the paper is organized as follows: In Section 2, we give some background on machine learning concepts in the context of sequence prediction. Section 3 is dedicated to the presentation of our AL-Net model. The model is further validated both in realistic conditions and against a state-of-the-art model in Section 4. The work is finally concluded in Section 5.

2 Background on Time Series Forecasting

One of the most used method for time series forecasting is the auto-regressive integrated moving average (ARIMA) [11]. ARIMA model however suffers from the assumption of linear relationship between the predictors and the target variable. In order to overcome this limitation, deep machine learning approaches are more and more used for time series forecasting. In the following sections, we introduce classical deep learning models that are applied in this context.

2.1 Multilayer Perceptron

Multilayer perceptrons (MLP) [12] is a promising tool for most classification and regression problems. In the case of regression, a MLP is trained on a set of data of the form (X, y) where :

- X is a vector of features
- y is the target

The model approximates a function f such that $f(X) = y$.

There exists many types of MLPs, each of them having its own specificities. For sequence to sequence prediction use cases, one of the most promising MLP type is the Long Short Term Memory Network.

2.2 Long Short Term Memory Network

Long short term memory networks (LSTM) [13] were introduced to overcome the long term dependency issue encountered by traditional recurrent neural networks (RNN). LSTMs have a chain structure similar to the one we observe in a standard RNN. The main difference that provides LSTMs with the ability to handle long term dependencies lies within the modules. Each module comprises a four-layers network called gates. Those gates are shown in Figure 1. Using this gates mechanism, an LSTM is able to select, edit, or delete persistent information in the long or short term.

In the following Section, we introduce our Auto-Lag Net model which embeds the contributions of this paper.

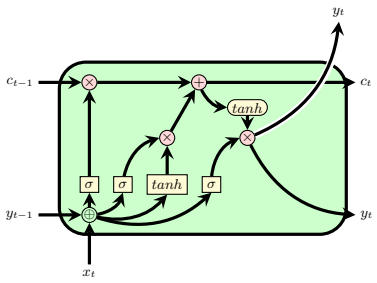


Fig. 1: Gates within an LSTM cell

3 Auto-Lag Net

In this section, we present how we iteratively construct our Auto-Lag Net architecture. The key difference between the first stage model and the final one is the dynamic lag factor selection through an attention mechanism. Auto-Lag Net uses an encoder-decoder architecture [14], [15]. The key idea is that the model is made up of several stacked layers separated into two groups. The first one called encoder reads the input in and produces a fixed sized representation. The second group named decoder is responsible for reading the encoded representation of the input and produces the target output.

3.1 Encoder

The encoder is composed of 4 layers. The first two layers are convolutional layers. In order to preserve the temporal order within the data, those layers are padded. The third encoder layer is a max pooling layer whose main purpose is to reduce data dimensionality. Those layers (*conv1d_1, conv1d_2, max_pooling_1d_1*) are shown in Figure 2. Convolutions and max pooling layers are image oriented layers. Traditionally, they are applied to $2D$ image data with 3 channels (rgb). Nevertheless, for a stride length that is less than the time stamp, it is natural to apply a one dimensional convolution to a time series as it evolves with the time. The last encoder's layer is an LSTM layer (see *lstm_1* in Figure 2).

3.2 Decoder

The decoder is made of a stack of 3 layers. The first is an LSTM layer identical to the encoder's last layer (*lstm_2* in Figure 2). The second and the third decoder's layers are two identical time distributed dense layers.

Most sequence to sequence models are auto-regressive, i.e they consume the last generated symbol to produce the next one [16], [17]. We can see from Figure 2 that our decoder takes two inputs in. The first input given by *lstm_1* is the encoder's output. The second one given by *input_2* is the previously generated symbol (it is the decoder hidden state denoted h_t), it is called the context.

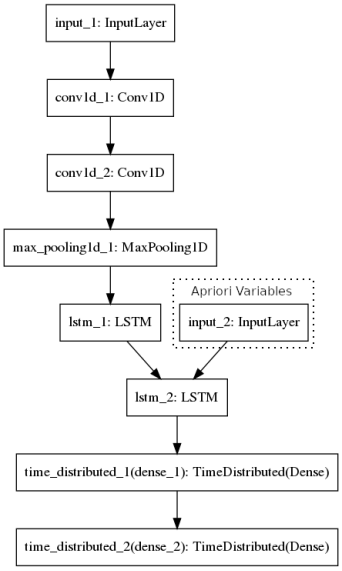


Fig. 2: The basic model architecture

3.3 Error Accumulation

Most auto-regressive machine learning models are trained with the *teacher forcing* method [18]. Teacher forcing uses the real output from the training data set at a time step t as input to produce the next output at time step $t + 1$. During the training, the model's loss is therefore evaluated on producing a single value at each time step. As noticed in [19], this presents two major issues:

1 **Exposure bias**. The *Exposure bias* is due to the fact that the model is trained to predict the future value of a time series given the previous ground truth value as input. The model is therefore never exposed to its errors during the training. At test time, the model iteratively produces a whole sequence from scratch by consuming at each step its own previous output to produce the next one. This causes the error to accumulate along the way.
2 **Train/Test metrics incoherence**. The *Train/Test metrics incoherence* issue is due to the fact that, the training loss is evaluated upon producing a single value at each time step whereas during testing, the model is evaluated on the whole produced sequence

As we will experimentally confirm in Section 4, the accuracy of the so constructed model suffers from error accumulation. To avoid the *Exposure bias* issue, some papers in the literature use the *beam search* heuristic [14], [20], [21]. Instead of using the model's output at a time step t to produce the output at time step $t + 1$, a *beam search*

decoder keeps at each step t a list of k best candidate outputs. The beam search heuristic produces great results in text related tasks were the output dictionary is discrete and finite.

To overcome both the *Exposure bias* and the *Train/Test metrics incoherence* issues in the case of real valued sequence to sequence prediction, we proceed as follows:

1 **No teacher forcing and a priori variables.** The model's output is never fed back to produce the next time step output. Instead, we feed the decoder with additional variables that describe the prediction window. Examples of such variables include the weather forecast, day of the week, month, whether it is holiday or not. Values for those variables are known at prediction time, we call them *a priori variables*. In this work, we define *a priori variables* as exogeneous variables whose future values are always known in advance. By doing this, we avoid the *Exposure bias* issue.

2 **Multi-step-ahead direct forecasting.** In [22], the authors examine two alternative approaches to auto-regressive models: Independent value prediction and Parameter prediction. While the parameter prediction methods does not suffer from error accumulation, it doesn't take into consideration the time dependence between values of the sequence.
The AL-Net model uses a multiple input multiple output (MIMO) [23] strategy to produce the whole target sequence at once. As shown in [24], this strategy presents many advantages. This approach solves the *Train/Test metrics incoherence* issue as the model's loss is evaluated during training on the whole predicted sequence. By doing so, we also avoid error accumulation.

3.4 Dynamic Lags

A lagged value is a delayed variable. Many time series forecasting models [25] take into account lagged values of the target as input. The intuition is that, there might exist a natural number k such that the value of a time series at a given time step $t + n$ is correlated to its value at time step $t - k$.

While numerous papers introduce sophisticated lag selection algorithms [7] [26], they all end up finding optimal observation windows. In this paper, we go a step further. We rely on the following intuition : All the values within an observation window do not account at the same time to produce every single value of the output sequence. As it is the case with problem involving neural image caption generation [4], a fixed representation of whole input observation window (the output of the encoder) is not optimal to produce every single value in the output sequence.

We therefore propose a dynamic lag selection mechanism. As illustrated in the heatmap of Figure 6, we can see how the dynamic lag selection module makes use of the above intuition. Indeed, it learns to focus on different locations of the observation window to produce specific values of the output sequence. This relies on the attention mechanism.

Attention mechanism was first introduced in 2014 [3] to overcome the issue in neural machine translation were the decoder has to produce the whole translated sequence out of a summarized version of the input sequence.

There exists several implementations of the attention mechanism in the literature. They are classified into two categories : global and local attention [3]. Our dynamic lag selection mechanism relies on an original modification of a global attention mechanism.

A global attention model takes two arguments. A vector $\overline{y} = (\overline{y_1}, \ldots, \overline{y_n})$ and a context vector c of length n. It computes a vector of length n, that is interpreted as the relevant parts of $\overline{y}$ with respects to the context c.

Specifically, at each time step t, it does the following:

- Using the decoder intermediary states h_t, a context vector c_t is computed.
- A concatenation layer is used to produce an attentional layer $\overline{h}$ with the following equation :

$$\overline{h} = \tanh(W_c[c_t h_t]). \tag{1}$$

- $\overline{h}$ is then fed to a softmax layer to predict next values.

Using the attention model terminology, our dynamic lag selection mechanism is implemented as follows:

- Context vector c_t
  The context vector is computed as follows:
  1 Since we do not produce the target sequence in an iterative way (see Section 3.3) we can't access the decoder's intermediary states. We set h_t to be the output of the a priori variables fed to an LSTM (*lstm_2* of Figure 3).
  2 c_t is obtained by taking a dot product over the encoder's output $\overline{y}$ and h_t (*dot_1* of Figure 3).

$$c_t = \overline{y}.h_t.$$

- The attention a_t.
  At any time step t, the attention produces relevant parts of an input $\overline{y}$ with respects to a given context c_t. We proceed as follows: The context c_t is first fed to a softmax layer (*activation_1* of Figure 3). Second, a dot product is taken over the output of that softmax layer and the output of the encoder (*dot_2* of Figure 3).

$$c'_t = softmax(c_t),$$

$$a_t = \overline{y}.c'_t.$$

- The attentional vector $\overline{h}$.
  We finally produce the attentional vector $\overline{h}$ with a concatenation layer (*concatenate_1* of Figure 3)[3], following Equation 1.

In Figure 3, our dynamic lag module is made up of layers *dot_1*, *activation_1*, *dot_2* and *concatenate_1*.

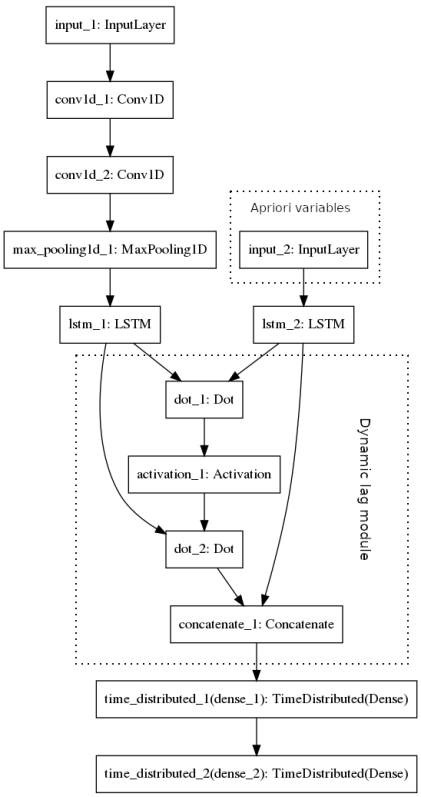


Fig. 3: The Auto-Lag net architecture with the dynamic lag selection module

4 Experiments

In this Section, we present the performance results of AL-Net on two real world data set. The first one is a simple time series use case while the second is a hierarchical time series one.

4.1 Electricity consumption

We applied our AL-Net on an electricity consumption use case. It is a unit commitment problem where the electricity provider needs to adapt the energy production with energy consumption. The amount of produced energy needs to be carefully chosen in order to satisfy the demand without surpassing it. Surpassing the demands results in financial losses since it is complicated to store energy [27]. In order to remain in realistic conditions, we configure the AL-Net to predict a whole day of future energy consumption so that the energy provider can adapt the next day's production. Since the time step is 30 minutes, our prediction window is then made of 48 values i.e we want to predict the next 48 values of the energy consumption.

4.1.1 Dataset We used the *Réseau de transport d'électricité (RTE) de France* data. This data set is the electricity consumption of France from January 1^{st} 2012 to July 1^{st} 2018 The time step is 30 minutes. We splitted the data into a sequential train test split in order to preserve data chronology. Train starts on January 1^{st} 2012 and ends on September 30^{the} 2016. The test set starts on October 1^{st} 2016 and ends on July 1^{st} 2018. Figure 4 shows the profile of the data set as well as the train/test split.

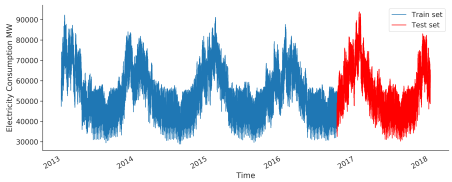
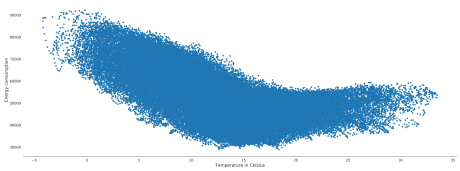


Fig. 4: RTE dataset splitted into train and test sets.

4.1.2 A priori Variable Selection As mentioned in Section 3.3, the AL-Net's output is never fed back as input to the model. We instead feed the encoder with explanatory variables called a priori variables. The purpose of those variables is to help the model to "understand" the target values. For example, a very high consumption value may be interpreted by the model as an outlier. However, an a priori weather variable might explain to the model that the encountered high value is due to a very low temperature at that time. The choice of the a priori variables is made by intuition. For example, Figure 5a shows that, generally the energy consumption decreases as the temperature increases. Similar observation is shown in Figure 5b with the type of the day. Therefore, we use the following a priori variables in our model : Week, Month, Temperature, Day type.



(a) Link effect relation between temperature and energy consumption.

(b) Day type against energy consumption

Fig. 5: Intuitions on the choice of a priori variables

4.1.3 Results In order to evaluate the AL-Net's performance, we used the root mean squared error (RMSE). This choice is justified by the fact that it penalizes high errors. This property is suitable as in our use case, large errors lead to huge mismatch between energy supply and demand. We used XGBoost with a priori input variables as baseline algorithm.

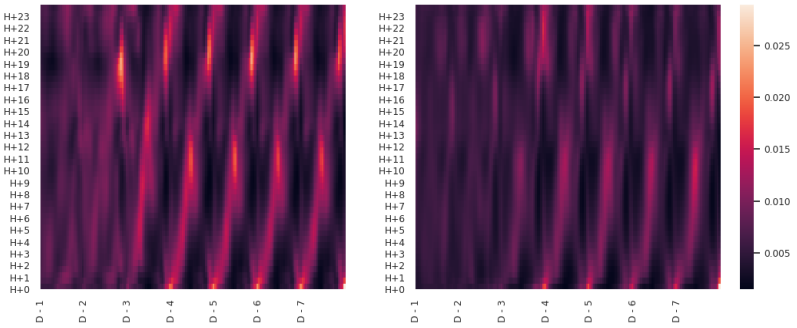
The results are presented in Table 1. The basic configuration of the model is the model without a priori variables and without dynamic lag selection. The fact that our base model is already more than 140% better than XGBoost shows that Al-Net is a good approach for time series forecasting problems. Iteratively adding day types and temperature as a priori variables takes the model performances from 2400 RMSE to 1247. This clearly shows the contribution of a priori variables to the model and thus validates our intuition. The last step further is the dynamic lag module. This again increases the performance of the model from 1247 RMSE to 991 i.e below 1000

Configuration	RMSE
XGBoost	5890
Basic	2400
+Day types	1650
+Temperature	1247
+Dynamic lag	991

Table 1: RMSE values for various configurations of the model

Figure 6 shows the heat-map of the dynamic lag selection module. The x-axis are the lagged values and the y-axis is the target 24-hours prediction window. The heat-map shows how the model dynamically selects lag values to compute each of the 48 target values. A hotter color indicates a higher contribution of the corresponding lag. Specifically, we observe from the activation of the attention on a winter day and on a summer day (Figure 6) how the selection of the lagged values dynamically changes. For example, to predict the energy consumption at $12PM$ on a winter day, the model focuses on lagged values around $12PM$ and $2PM$ of day-3 up til day-7. While on a summer day it instead focuses on lagged values around $10AM$ and $12PM$.

4.1.4 Model Validation on State-Of-The-Art Results In [7] the authors present an LSTM based sequence to sequence model to predict energy consumption with the RTE data set. Their model rely on feature selection and a genetic algorithm for lag selection. In order to evaluate our model against their state-of-the art performing model, we put ourselves in the unrealistic situation were the energy supplier needs only the next time step forecast of the consumption. We therefore set our prediction window to be a single time step. After feature selection and hyper-parameter tuning using their genetic algorithm, the RMSE score of their best model is 341.40. Using our AL-Net model, we surpass this score by up to $\mathbf{15}\%$ without model tuning. More precisely, on that very same problem configuration, AL-Net obtains a $\mathbf{295.97}$ RMSE score.



(a) Typical winter day (b) Typical summer day

Fig. 6: Dynamic lag selection heat map

In order to compare AL-Net to [7]'s LSTM Metrics Optimal Time Lags Model (LMOTLM) on different time horizons, we train AL-Net to predict a whole day prediction window. Among the test set predictions, we randomly selected 100 single value predictions at various time horizons. The results are given in Table 2.

Time Horizon	LMOTLM	AL-Net
2 weeks	339	221
2-4 weeks	258	245
2-3 Months	294	240
3-4 Months	275	290

Table 2: Performance comparison on various time horizons

Overall, AL-Net is 17% better than [7]'s model. This results is impressive considering the fact that AL-Net was trained to predict a whole day. Further, we observe that these results are better than the 295.97 RMSE score achieved by AL-Net when trained to output a single target value. This observation definitely validates that, the architecture of AL-Net is well adapted to multi-step ahead direct forecasting. This is due to the fact that, by construction, the AL-Net gets more a priori informations (for the whole prediction windows) in multi-step ahead direct forecasting mode than in single-step ahead forecasting mode.

4.2 Rossmann Store Sales

The Rossmann Store Sales is a hierarchical time series use case that was subject to a kaggle competition featuring more than 3000 participants [10]. We fitted the publicly available data set to the AL-Net to evaluate its performances on hierarchical time series.

4.2.1 Data set and results The data set of this use case is the historical data of 1115 Rossman stores located across Germany, and the task is to predict 6 weeks of daily sales. Variables of the data set include *Store* and *StoreID*. We use those non temporal features to specify the group to which belong a time series. In this use case, AL-NET is used alone i.e not embedded in an ensemble method, and yet obtains a root mean squared percentage error (RMSPE) $\mathbf{0.13826}$. When we exclude the aberrant RMSPE score of 2798997865347.38 obtained by a participant of the challenge, the remaining public leaderboad scores lies within the interval from 0.08932 to 1. AL-Net is thus by 35.39% worse than the best score, but more than 1000% better than the worst score in the public challenge leaderboard. Those satisfactory results show that the one-shot approach of AL-Net is competitive in hierarchical time series use cases.

5 Conclusion

In this paper, we presented a LSTM based sequence to sequence model in the context of real valued sequence prediction. The contribution of the paper is two-fold. First we propose a model architecture that overcomes the accumulating error problem encountered in classical sequence to sequence models. Second, we propose a dynamic lag selection mechanism. The proposed dynamic lag selection mechanism requires no knowledge of the data and can therefore be applied to any time series or hierarchical time series data set. Experimental results shows that our AL-Net model outperforms state-of-the-art models. Our model also shows promising performances on a realistic configuration and can therefore be deployed in real life.

Acknowledgement

We would like to thank the reviewers for their thoughtful comments towards improving our paper and our colleague Lucas Perret for the great production code quality.

References

1. Ekaterina Arafailova, Nicolas Beldiceanu, Rémi Douence, Mats Carlsson, Pierre Flener, María Andreína Francisco Rodríguez, Justin Pearson, and Helmut Simonis. Global constraint catalog, volume ii, time-series constraints. *arXiv preprint arXiv:1609.08925*, 2016.
2. Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
3. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
4. Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.
5. Gilles Madi Wamba, Yunbo Li, Anne-Cécile Orgerie, Nicolas Beldiceanu, and Jean-Marc Menaud. Cloud workload prediction and generation models. In *SBAC-PAD*, pages 89–96. IEEE, 2017.
6. Narayana Prasad Padhy. Unit commitment-a bibliographical survey. *IEEE Transactions on power systems*, 19(2):1196–1205, 2004.

7. Salah Bouktif, Ali Fiaz, Ali Ouni, and Mohamed Serhani. Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies*, 11(7):1636, 2018.
8. Daniel Kosiorowski, Dominik Mielczarek, Jerzy Rydlewski, et al. Forecasting of a hierarchical functional time series on example of macromodel for day and night air pollution in silesia region: A critical overview. *arXiv preprint arXiv:1712.03797*, 2017.
9. Rob J Hyndman, Roman A Ahmed, George Athanasopoulos, and Han Lin Shang. Optimal combination forecasts for hierarchical time series. *Computational Statistics & Data Analysis*, 55(9):2579–2589, 2011.
10. Rossmann store sales. https://www.kaggle.com/c/rossmann-store-sales/. Accessed: 2019-01-07.
11. George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
12. Richard Lippmann. An introduction to computing with neural nets. *IEEE Assp magazine*, 4(2):4–22, 1987.
13. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
14. Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
15. Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
16. Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
17. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
18. Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.
19. Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.
20. Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*, 2017.
21. Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*, 2016.
22. Haibin Cheng, Pang-Ning Tan, Jing Gao, and Jerry Scripps. Multistep-ahead time series prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 765–774. Springer, 2006.
23. Francisco Zamora-Martínez, Pablo Romeu, Paloma Botella-Rocamora, and Juan Pardo. Towards energy efficiency: Forecasting indoor temperature via multivariate analysis. *Energies*, 6(9):4639–4659, 2013.
24. Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert systems with applications*, 39(8):7067–7083, 2012.
25. Luke Keele and Nathan J Kelly. Dynamic models for dynamic theories: The ins and outs of lagged dependent variables. *Political analysis*, 14(2):186–205, 2006.
26. Gustavo HT Ribeiro, Paulo SG de Mattos Neto, George DC Cavalcanti, and Ing Ren Tsang. Lag selection for time series forecasting using particle swarm optimization. In *IJCNN*, pages 2437–2444, 2011.
27. Piyasak Poonpun and Ward T Jewell. Analysis of the cost per kilowatt hour to store electricity. *IEEE Transactions on energy conversion*, 23(2):529–534, 2008.