Deep Learning for Energy Time Series

Kai Schmieder
Motivation

- Buildings are major energy consumer worldwide (Pérez-Lombard et al. 2008)

- Minimize the energy wastage and making power generation and distribution more efficient by intelligent control decisions (Marino et al. 2016, p. 7046)

- Load forecasting crucial for mitigating uncertainties of the future
Motivation

- Deep Learning proved to be useful in manifold fields
  - Computer vision
  - Speech and audio processing
  - Natural language processing
  - Robotics
  - Bioinformatics and chemistry
  - Finance
  - ...

(Goodfellow et al. 2016, p.8f)
Agenda

- Foundation
- Long Short-Term Memory for Energy Time Series
- Evaluation
- Discussion
- Conclusion
Energy Time Series

- Aggregate level and building level forecasting categorized by
  - Short-term → one hour to one week
  - Medium-term → one week to one year
  - Long-term → ranges longer than one year
  (Mocanu et al. 2016, p. 91)

- Two main approaches to forecast Energy Time Series
  - Statistical and Machine Learning based models
  - Physical Principles based models
  (Mocanu et al. 2016, p. 91; Marino et al. 2016, p. 7046)

  → here: Short-term & Statistical and Machine Learning based models
Deep Learning

- Central goal of Machine Learning
  - Learn useful representations of input data
  - That get us closer to expected output

- Deep Learning
  - Specific subfield of Machine Learning
  - Idea of successive layers of representations
  - Depth: how many layers contribute to a model of the data
  - Layered representations (almost always) learned via neural networks

(Chollet 2018, p. 6ff)
Overview of relevant Deep Architectures

- **Deep Forward Networks**: e.g. Multi-Layer-Perceptrons
- **Convolutional Neural Networks**: e.g. AlexNet for image classification
- **Recurrent Neural Networks**: e.g. Long Short-Term Memory
- **Deep Generative Models**: e.g. Boltzmann Machines
- **Autoencoders**: e.g. For Feature Extraction
- **Mixes of architectures**: e.g. Convolutional Neural Networks with Recurrent Neural Network

Figure 2: Own representation based on Goodfellow et al., 2016, p. 11

<table>
<thead>
<tr>
<th>Foundation</th>
<th>Long Short-Term Memory for Energy Time Series</th>
<th>Evaluation</th>
<th>Discussion</th>
<th>Conclusion</th>
</tr>
</thead>
</table>
Recurrent Neural Networks

- Family of neural networks for processing sequential data
- Basic principle
  - Each member of the output is a function of the previous member of the output
  - Unrolled Recursive Neural Network

(Goodfellow et al. 2016, p. 363f)

LONG SHORT-TERM MEMORY FOR ENERGY TIME SERIES
Building Energy Load Forecasting using Deep Neural Networks

Daneil L. Marino, Kasun Amarasinghe, Milos Manic
Department of Computer Science
Virginia Commonwealth University
Richmond, Virginia
marinodl@vcu.edu, amarasinghek@vcu.edu, mikos@vcu.edu

Abstract—Ensuring sustainability demands more efficient energy management with automated energy usage. Therefore, the power grid in the future should provide an unprecedented level of flexibility in energy management. To that end, intelligent decision making requires accurate predictions of future energy demands, both at aggregate and individual site level. Thus, energy load forecasting have received increased attention in the recent past. However, it has proven to be a difficult problem. This paper presents a novel energy load forecasting methodology based on Deep Neural Networks, specifically Long Short-Term Memory (LSTM) architecture. The presented work investigates two different architectural designs: 1) standard LSTM and 2) LSTM-based Sequence-to-Sequence (S2S) architectures. Both architectures were implemented on a benchmark dataset of electricity consumption data from one residential customer. Both architectures were trained and tested on one hour and one-minute time-step resolution datasets. Experimental results showed that the standard LSTM failed on one-minute resolution data while performing well on one-hour resolution data. It was shown that the S2S architecture is capable of modelling long-term dependencies and can deliver competitive results. Therefore, the presented methods produced comparable results with other deep learning methods for energy forecasting in literature.

Keywords—Deep Learning; Deep Neural Networks; Long-Short-Term Memory (LSTM) Architecture; Energy Load Forecasting

I. INTRODUCTION

Buildings are identified as a major energy consumer worldwide, accounting for 20%-40% of the total energy production [15]. In addition to being a major energy consumer, buildings are shown to account for a significant portion of energy savings as well [4]. As energy savings poses a threat to sustainability, making buildings energy-efficient is extremely crucial. Therefore, in making building energy consumption more efficient, it is necessary to have accurate predictions of its future energy consumption.

At the grid level, in order to maintain the energy usage and making the power generation and distribution more efficient, the future of the power grid is moving to a new paradigm of smart grids [5, 6]. Smart grids are promising, unprecedented flexibility in energy generation and distribution [7]. In order to provide that flexibility, the power grid has to be able to dynamically adapt to the changes in demand and efficiently distribute the generated energy from the various sources such as renewables [5]. Therefore, intelligent control decisions should be made continuously at aggregate level as well as modular level in the grid. In achieving this goal and ensuring the reliability of the grid, the ability of forecasting the future demands is important [6, 9]. Further, demand or load forecasting is crucial for mitigating uncertainties of the future [6]. In that, individual building level demand forecasting is crucial as well as forecasting aggregate loads. In terms of demand response, building level forecasting helps carry out demand response locally since the smart grids have made the acquisition of energy consumption data at building and individual site level feasible. This data driven and statistical forecasting models are made possible [7].

At the aggregate level and building level, load forecasting can be viewed in three different categories: 1) Short-term (1-24 hours), 2) Medium-term (1-30 days), and 3) Long-term (6 months-5 years). It has been demonstrated that the load forecasting is a hard problem and in that, individual building level load forecasting is even harder than aggregate load forecasting [8]. Therefore, the work presented in this paper investigates a methodology for performing energy load forecasting: 1) Physics principles based models and 2) Statistical and machine learning based models. Focus of the present work is on the second category of statistical load forecasting. In [7], the authors use Artificial Neural Network (ANN) assembly to perform the building level load forecasting. ANNs have been explored to detect the purpose of all three categories of load forecasting [9, 10]. In [11], the authors use a support vector machines based regression model coupled with empirical mode decomposition for long-term load forecasting. In [12], the authors model individual household electricity loads using sparse coding to perform medium term load forecasting. In the interest of brevity, not all methods in literature are introduced in the paper. For surveys of different techniques used for load forecasting, readers are referred to [9], [10], [11] and [13]. Despite the enormous research carried out in the area, individual site level load forecasting remains to be a difficult problem.

Therefore, the work presented in this paper investigates a deep learning based methodology for performing individual household energy load forecasting. The methodology is composed of multiple layers to learn representations in data. The use of multiple layers allow the learning process to be carried out with multiple layers of abstraction. A comprehensive overview and a review of deep learning methodologies can be

Figure 4: Marino et al., 2016, p. 7046

Short abstract of Marino et al. (2016)

- Application of two architectural variations of Long Short-Term Memory
- Stacked Long Short-Term Memory
- Sequence-to-Sequence Long Short-Term Memory

(Marino et al. 2016)

Experiments on benchmark dataset of electricity consumption for a single residential customer (“Individual household electric power consumption”) (Dheeru and Taniskidou 2017)
Overview of presented Long Short-Term Memory architectures

Stacked Long Short-Term Memory

Sequence-to-Sequence Long Short-Term Memory

Figure 5: Own representation based on Marino et al., 2016, p. 7047

Figure 6: Own representation
Long Short-Term Memory

- Type of Recurrent Neural Network introduced by Sepp Hochreiter and Jürgen Schmidhuber (1997)
  - Great success in various applications (Goodfellow et al. 2016, p. 363f)
  - Solves vanishing gradient problem

Key concepts

- Cell State
- Gates
  - Input
  - Forget
  - Output

Figure 7: Olah, 2015, http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-chain.png, Retrieved on 03.12.2018
Long Short-Term Memory Cells

0 Cell State

1 Forget Gate

Figure 8: Olah, 2015,
http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-C-line.png, Retrieved on 03.12.2018

Figure 9: Adapted from Olah, 2015,
http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-f.png, Retrieved on 03.12.2018

\[ f_t = \sigma \left( W_f \cdot [h_{t-1}, x_t] + b_f \right) \]
Long Short-Term Memory Cells

2 Input Gate

\[ i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \]
\[ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \]

Figure 10: Adapted from Olah, 2015, 
http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-i.png, Retrieved on 03.12.2018

3 Update Cell State

\[ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \]

Figure 11: Adapted from Olah, 2015, 
http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-C.png, Retrieved on 03.12.2018
Long Short-Term Memory Cells

4 Output Gate

\[
\begin{align*}
o_t &= \sigma (W_o [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \times \tanh (C_t)
\end{align*}
\]

Figure 12: Adapted from Olah, 2015, http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-o.png, Retrieved on 03.12.2018
Stacked Long Short-Term Memory architecture

- Stacked architecture
  - Stack multiple Long Short-Term Memory Cells into a multi-layer architecture
  - Dense layer

- Input vector: \( i_t = [ y_{t-1}, \text{day}_t, \text{day-of-week}_t, \text{hour}_t ] \)
  - Use also more than one time step as “context”

- Output vector: \( \hat{y}_t \)

(Marino et al. 2016)
Overview of presented Long Short-Term Memory architectures

Stacked Long Short-Term Memory

Sequence-to-Sequence Long Short-Term Memory

Figure 5: Own representation based on Marino et al., 2016, p. 7047

Figure 6: Own representation
The idea of Sequence-to-Sequence Learning

- Introduced by Sutskever et al. (2014) to map sequences of different lengths

- Comprised of two sub-models
  - Encoder
    - Convert input sequences of variable length and
    - Encode them in a fixed length vector, which is then used as input state for the decoder
  - Decoder
    - Generates an output sequence of fixed length
    (Marino et al. 2016, p. 7048)

- Applications in speech recognition, machine translation and question answering (Goodfellow et al. 2016, p. 385)
**Sequence-to-Sequence**

**Long Short-Term Memory architecture**

- **Encoder**
  - Input vector: \( i_t = [ y_{t-1} \ \text{day}_t \ \text{day-of-week}_t \ \text{hour}_t ] \)
  - “Output”: Cell State

- **Decoder**
  - Input vector: \( f_t \)
  - Output vector: \( \hat{y}_t \)

(Marino et al. 2016)
EVALUATION
Experiments in Marino et al. (2016)

Data
- “Individual household electric power consumption” (Dheeru and Taniskidou 2017)
- Aggregation Level
  - 1 minute (original)
  - 1 hour
- Training set: 3 years; Test set: 1 year

Model
- Stacked LSTM
- Sequence-to-Sequence LSTM

Horizon
- One Step
- 60 Steps

Evaluation and results
- Stacked LSTM performs well with inputs further in the past for hourly data
- Sequence-to-Sequence LSTM performs well in both datasets
- Comparable results to Mocanu et al. (2016)
Implementation and hardware resources

- Jupyter Notebooks at https://github.com/nicoleludwig/EnergyInformatics

- Implementation with Python
  - Pandas
  - NumPy
  - scikit-learn
  - Keras with TensorFlow as backend
  - Chartify

- Hardware resources for training
  - GPU: NVIDIA Tesla V100*

* Made possible by Andreas Bartschat (IAI), thanks again! 😊
Data for evaluation

- Energy consumption data of KIT Campus Nord
  - Building 124
  - 15 minutes resolution
  - Training set of three years
    (Mid May 2012 until Mid May 2015)
  - Test set of one year
    (Mid May 2015 until Mid May 2016)

Figure 13: https://www.kit.edu/downloads/Campus-Nord.pdf, Retrieved on 05.01.2019
Additional time series approaches

- Stationarity
  - “A stationary time series is one whose properties do not depend on the time at which the series is observed.” (Hyndman and Athanasopoulos 2018)
  - Implementation with differencing to remove effects

  - Min-Max-Scaling
    - Transform features by scaling each feature to a given range
    - Implementation with range [0, 1]
  - Standardization
    - Standardize features by removing the mean and scaling to unit variance (z-transformation)
    - Implementation with mean = 0 and standard deviation = 1
### Overview of evaluation scenario characteristics

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of Aggregation</td>
<td>15 minutes</td>
</tr>
<tr>
<td>Stationarity</td>
<td>Stationary</td>
</tr>
<tr>
<td>Normalization</td>
<td>None</td>
</tr>
<tr>
<td>Models</td>
<td>Stacked Long Short-Term Memory</td>
</tr>
<tr>
<td>Horizon</td>
<td>One step</td>
</tr>
</tbody>
</table>

→ 72 scenarios per building
Evaluation metric

- Root mean squared error (RMSE)

\[
RMSE = \sqrt{\frac{1}{T \cdot n_v} \sum_{t=1}^{T} \sum_{i=1}^{n_v} (v_{i,t} - \hat{v}_{i,t})^2}
\]

- \( n_v \rightarrow \) number of steps in horizon
- \( T \rightarrow \) total number of steps predicted into the future
- \( v_{i,t} \rightarrow \) real values for time-step \( t \)
- \( \hat{v}_{i} \rightarrow \) value predicted by the model for time-step \( t \)

(Mocanu et al. 2016, p. 95)
Baseline models

- Naïve Forecast
  - Value of the last week on the same day of the week at the same time
  - Prediction for timestep $t$: $(t - 672 / t - 168)$

- Ordinary Regression
  - Regression of the respective load time series with the following regressors
    - Load before one day $(t - 96 / t - 24)$
    - Load before two days $(t - 192 / t - 48)$
    - Load before one week $(t - 672 / t - 168)$
    - Dummy variables for weekend, month, hour and minute (only for 15 minutes)
## Results for Building 124

<table>
<thead>
<tr>
<th>Level of Aggregation</th>
<th>Stationary</th>
<th>Normalization</th>
<th>RMSE per time horizon</th>
<th>Ordinary Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Stacked-LSTM</td>
<td>S2S-LSTM</td>
</tr>
<tr>
<td>15 Minutes</td>
<td>No</td>
<td>None</td>
<td>(0.015; 0.853; 0.893)</td>
<td>(1.474; 1.496; 1.491)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min-Max</td>
<td>(0.01; 0.792; 0.865)</td>
<td>(1.482; 1.473; 1.491)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard</td>
<td>(0.011; 0.816; 0.868)</td>
<td>(1.475; 1.489; 1.494)</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>None</td>
<td>(0.007; 1.193; 1.726)</td>
<td>(0.427; 1.755; 2.022)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min-Max</td>
<td>(0.016; 1.039; 3.196)</td>
<td>(0.427; 1.761; 2.093)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard</td>
<td>(0.008; 1.177; 1.693)</td>
<td>(0.427; 1.755; 2.023)</td>
</tr>
<tr>
<td>1 Hour</td>
<td>No</td>
<td>None</td>
<td>(0.021; 0.742; 0.871)</td>
<td>(1.453; 1.449; 1.452)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min-Max</td>
<td>(0.026; 0.712; 0.81)</td>
<td>(1.433; 1.447; 1.453)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard</td>
<td>(0.03; 0.73; 0.864)</td>
<td>(1.443; 1.456; 1.452)</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>None</td>
<td>(0.008; 0.839; 1.291)</td>
<td>(0.558; 1.693; 1.964)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Min-Max</td>
<td>(0.049; 0.885; 1.899)</td>
<td>(0.558; 1.695; 2.015)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard</td>
<td>(0.009; 0.856; 1.208)</td>
<td>(0.558; 1.7; 1.964)</td>
</tr>
</tbody>
</table>

**Legend**
Root Mean Squared Error (One Step; One Day; One Week) | LSTM – Long Short-Term Memory | S2S – Sequence-to-Sequence
Remarks on results

- Technical and practical insights
  - Handling time shift and some missing days in 2012
  - Data pre-processing pipeline
    - Traditional time series analysis: 2D [samples, features]
    - Deep Learning: 3D tensor [samples, time-steps, features]
  - Distinction between one-step and multi-step forecasts
  - Parameters

- Time for training per Long Short-Term Memory model approx. 6.5 hours for all scenarios
DISCUSSION
Critical reflection on Marino et al. (2016)

Missing parameter configurations
- Training batch and epochs
- Norm clipping
- Dropout
- Activation function at Dense layer
- ...

Comparable results with Mocanu et al. (2016)
- In Mocanu et al. (2016) seven experiments with different resolutions and time horizons on the same dataset (Dheeru and Taniskidou 2017)
- Comparable result in one of seven scenarios → Results of other scenarios?
<table>
<thead>
<tr>
<th>Paper Reference</th>
<th>Deep Architecture</th>
<th>Deep Convolutional Neural Networks</th>
<th>Recurrent Neural Networks</th>
<th>Deep Generative Networks</th>
<th>Auto-encoders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amarasinghe et al. 2017</td>
<td></td>
<td>CNN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gensler et al. 2016</td>
<td></td>
<td>LSTM*</td>
<td></td>
<td></td>
<td>AE*</td>
</tr>
<tr>
<td>He 2017</td>
<td></td>
<td>CNN*</td>
<td>LSTM*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heghedus et al. 2018</td>
<td></td>
<td></td>
<td>GRU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jarábek et al. 2017</td>
<td></td>
<td></td>
<td>S2S-LSTM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Li et al. 2017</td>
<td></td>
<td>ELM*</td>
<td></td>
<td></td>
<td>SAE*</td>
</tr>
<tr>
<td>Marino et al. 2016</td>
<td></td>
<td></td>
<td>Stacked- &amp; S2S-LSTM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mocanu et al. 2016</td>
<td></td>
<td></td>
<td></td>
<td>RNN</td>
<td>CRBM &amp; FCRBM</td>
</tr>
<tr>
<td>Ryu et al. 2016</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RBM</td>
</tr>
<tr>
<td>Voß et al. 2018</td>
<td></td>
<td></td>
<td></td>
<td>WaveNet</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:** * - Combined Approach | CNN – Convolutional Neural Networks | (F)(C)RBM – (Factored) (Conditional) Restricted Boltzmann Machine | ELM – Extreme Learning Machine | GRU – Gated Recurrent Unit | LSTM – Long Short-Term Memory | RNN – Recurrent Neural Network | S2S – Sequence-to-Sequence | (S)AE – (Stacked) Autoencoder
CONCLUSION
Summary

Achievements
- Long Short-Term Memory as one Deep Learning method for Energy Time Series Forecasting
- Overview of current Deep Architectures for Energy Time Series
- Implementation and evaluation of
  - Stacked Long Short-Term Memory and
  - Sequence-to-Sequence Long Short-Term Memory based on Marino et. al. (2016)

Summarised results
- Stacked Long Short-Term Memory outperformed baseline models for hourly data in one-step and one-day predictions
- Good performance of Sequence-to-Sequence Long Short-Term Memory could not be confirmed (without hyperparameter optimization)
Outlook

Deep Learning
- Hyperparameter optimization
  - Grid Search
  - Random Search
  - Bayesian Optimization, etc.
- Variations of Long Short-Term Memory

Time Series
- Verify results with more buildings
- “Real” univariate Time Series Forecast
- Multivariate Time Series Forecast
Questions
Bibliography (1/5)


Bibliography (2/5)

Bibliography (3/5)


Bibliography (4/5)


APPENDIX AND BACKUP
How does deep learning basically work?

Source: Own representation based on Chollet (2018, p.11)
Activation functions

- **Sigmoid (σ)**
  - \( \sigma(t) = \frac{1}{1 + e^{-t}} \)
  - Value range \([0; 1]\)

- **Tangens hyperbolicus (tanh)**
  - \( \tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}} \)
  - Value range \([-1; 1]\)

Source:

Source:
Details: Stacked Long Short-Term Memory

- Objective function $L = \sum_{t=1}^{M} (y[t] - \hat{y}[t])^2$

- Unrolling implemented with M=50

- Optimizer Adam

- Training with Backpropagation Through Time

- Optimization with Norm clipping

Figure 5: Own representation based on Marino et al., 2016, p. 7047
Details: S2S Long Short-Term Memory

- Similar setup as Stacked Long Short-Term Memory

- Encoder objective function
  \[ L_E = \sum_{t=1}^{M} (y[t] - \hat{y}[t])^2 \]

- Decoder objective function
  \[ L_D = \sum_{t=M+1}^{T} (y[t] - \hat{y}[t])^2 \]

- Optimization with Norm clipping

- Regularization with Dropout

Figure 6: Own representation
Backpropagation Through Time for S2S LSTM

Encoder

$\hat{y}(1)$
$\hat{y}(2)$
$\hat{y}(M)$

$y[0]$, $f[1]$

$LSTM$ Layer 2


$LSTM$ Layer 1

$y[M-1], f(M)$

Decoder

$L_2$ Error

$\hat{y}(M+1)$
$\hat{y}(M+2)$
$\hat{y}(M+3)$

$y[M+1]$

$LSTM$ Layer 2

$y[M+2]$

$LSTM$ Layer 1

$y[M+3]$

$LSTM$ Layer 1

$y[T]$

$LSTM$ Layer 1

$y[T]$

$LSTM$ Layer 1

$LSTM$ Layer 2

$LSTM$ Layer 2

$LSTM$ Layer 2

$LSTM$ Layer 2