Estimation of the Probability of Successful Transmission of Uplink Messages in LoRaWAN

Ante Lojić Kapetanović

University of Split, Split, Croatia

May 15, 2019

This report is a short overview of applying forecasting methods on data from real LoRaWAN deployment in Svebølle, Denmark. The main idea is to examine whether there is a possibility of predicting successful transmission of LoRa messages for the given future time interval. Proposed model for prediction is Long Short-Term Memory (LSTM), an artificial recurrent neural network (RNN) architecture used in the field of deep learning.

1 Introduction

TBA

2 Network Topology

LoRaWAN - Long Range Wide Area Network - low power, wide area media access protocol designed to wirelessly connect 'things' to the internet. LoRaWAN is useful in IoT deployment, specially in smart environments where there are thousands of end-devices (i.e. sensors) that require minimal bandwidth and minimal energy.

The network architecutre is deployed in a star-of-stars topology in which gateways relay messages between end-devices and a central network server [3]. Toplogy overview is shown in Fig. 1.

End-device is transmitting data to base station using LoRa technology, which means that data is modulated using CSS technique and transmitted using 868MHz RF, also it is worth mentioning that data is encrypted using AES128 in Counter mode (CTR). If the packet's FPort is set to 0 then the NwkSKey is used, otherwise the AppSKey is used. An important feature of all messages in LoRa is that the counters for sent (FCntUp) and received (FCntDown) messages are maintained by the Node and Network Server, and that these counters never repeat [6]. Base stations relay messages from end-nodes to network server using a back-haul wireless network with higher frequency (often 2.4GHz) or using wired connection (Ethernet, optics). Network server decodes the packets received from base station and performs security checks and adaptive data rate so it can generate the packets that should be sent back to end-devices [3].

Observed LoRaWAN network model deployed in Svebølle is consisted of a single base station which communicates with few hundreds of different end-devices; network topology is quite simple and shown on Fig. 2.

Data was collected during the period of 4 months and 14 days, during which approximately 689 thousands measurements were generated and captured. There is only one base station and, judgdin by the exploration of the data set, 251 end-devices in the network. Measurements were carried out on the base station side and there are more than few relevant labels:

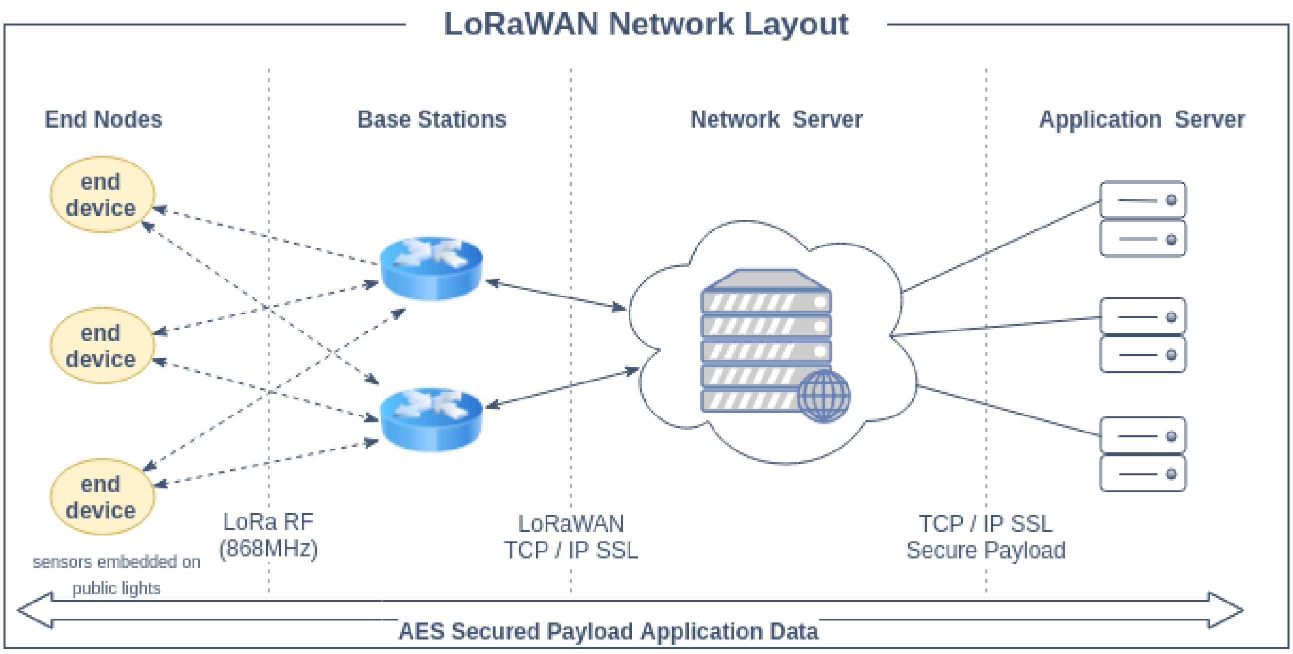


Figure 1: *LoRaWAN Network Layout*

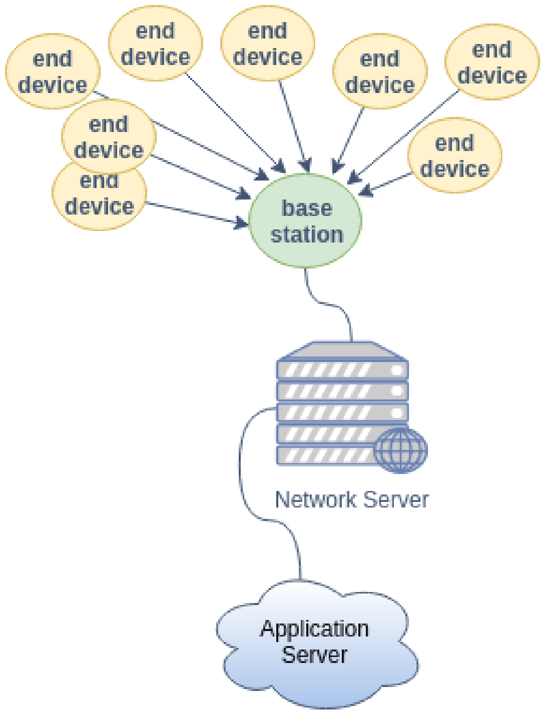
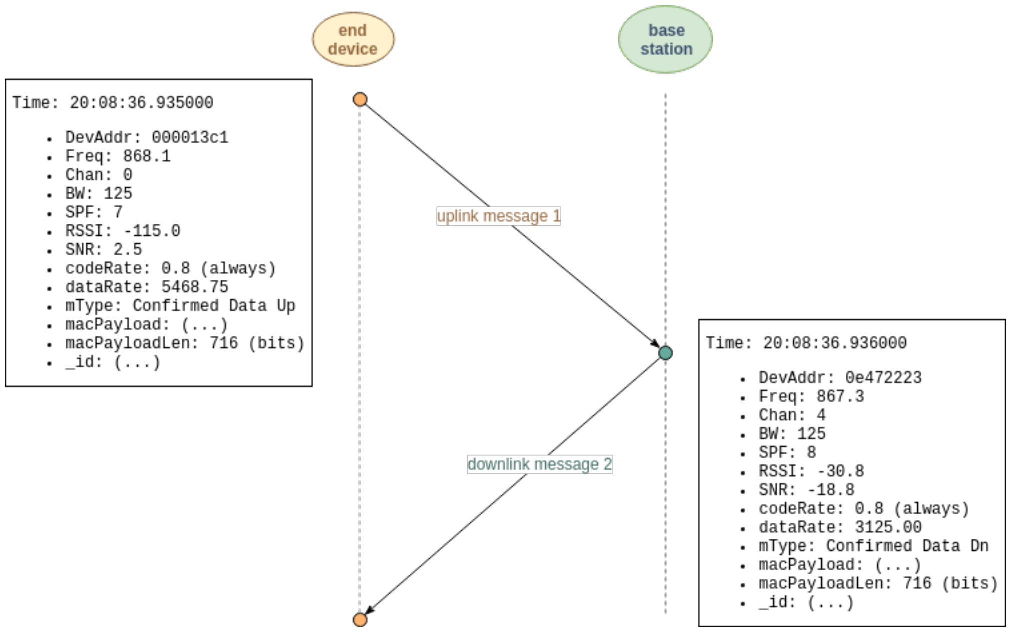


Figure 2: *Overview of Svebølle LoRaWAN network*



```
Time: 20:08:36.935000

  • DevAddr: 000013c1
  • Freq: 868.1
  • Chan: 0
  • BW: 125
  • SPF: 7
  • RSSI: -115.0
  • SNR: 2.5
  • codeRate: 0.8 (always)
  • dataRate: 5468.75
  • mType: Confirmed Data Up
  • macPayload: (...)
  • macPayloadLen: 716 (bits)
  • _id: (...)
```

```
Time: 20:08:36.936000

  • DevAddr: 0e472223
  • Freq: 867.3
  • Chan: 4
  • BW: 125
  • SPF: 8
  • RSSI: -30.8
  • SNR: -18.8
  • codeRate: 0.8 (always)
  • dataRate: 3125.00
  • mType: Confirmed Data Dn
  • macPayload: (...)
  • macPayloadLen: 716 (bits)
  • _id: (...)
```

Figure 3: *Communication between end-device and base station in LoRaWAN network in Svebølle*

- Time - basic datetime format to microsecond precision
- DevAddr - device address written as HEX string
- Freq - carrier signal radio frequency
- Chan - channel number
- BW - bandwidth
- SPF - spreading factor
- RSSI - received signal strength indication
- SNR - Signal-to-Noise ratio
- CR - code rate
- DR - data rate
- crcStatus - cyclic redundancy check
- mType - type of MAC message
- macPayload - part of transmitted data that is the actual intended message
- id - packet identification

All of these indicators are described in details in [2].

Considering the data, simple communication model of this LoRaWAN network could be deduced and is shown in Fig. 3

3 Introducing the Idea of ANN

Measurements in data set were not structured as time-series data, only times when transmission of LoRa message actually happend in uplink direction are captured and stored. Time series data is a series of data points indexed in time order and taken at successive equally spaced points in time. A time series is generally described through parameters such as trend, seasonality, irregularity and cyclicallity [1].

In order to create time series data, the only important label is wheter the end-device has successfully transmitted the message to the gateway or not. For each time point (second-precision) there is activity flag assigned. If the device was active for observed time point, activity flag is 1 otherwise activity flag is 0. Since the measurements were carried out through almost 5 month period, newly created data set was huge and as such perfect candidate for applying deep learning methods to perform forecasting of future time points.

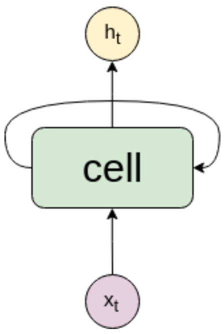


Figure 4: *RNN simple overview - feedback provides that the ouput (or a hidden layer) of the network, together with the new input at the current timestamp, is taken as input*

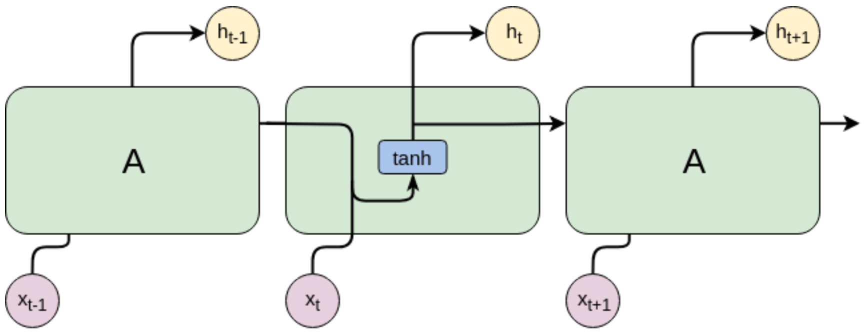


Figure 5: *RNN are formed in a chain-like nature what makes them applicable for predicting future events of sequence data*

Artificial Neural Networks, or ANN for short, are modeling the intelligence of human brain: recognizing regularities and patterns of the given data, learning from past experience and providing inference on the output [1]. Why are ANNs such good predictors for time series type of data? There are couple of reasons:

Data-driven and self-adaptive There is no need to make *a priori* assumption about statistaical distribution of the data before feeding ANN with the data [1].

Non-linearity ANNs are perfect for modeling data with non-obvious patterns [8].

Universal functional approximators Any contionous function can be approximated to any accuracy [5].

Even though vanilla ANNs are very powerful tool to create predictions based on historical events, they lack persistence. Recurrent Neural Networks (RNNs) are trying to solve this problem implementing loops and allowing information to persist, overview of RNN is shown in Fig. 4. RNN looks at the input x_t and outputs a value h_t, a loop allows information to be passed from one step of the network to the next. Unpacked RNNs have the form of a chain of repeating nodes defined by very simple structure, Fig. 5 [7].

RNNs are using gradient descent (first-order iterative optimization algorithm, usually used for finding the minimum of a function) as a way to minimize the error caused by changing each weight in proportion the derivative of the error with respect to that weight. Problem with this approach is error gradient vanishing which is causing the RNN to become unable to learn to connect the information if the relevant data is devided by large gap. The problem of long-term dependencies is solved by applying Long Short-Term Memory network, LSTM for short, which are introduced by the Sepp Hochreiter and Jurgen Schmidhuber in 1997. in the [4]. They proposed a model that is capable of remembering relevant information for long period of time and avoiding the long-term dependency issues.

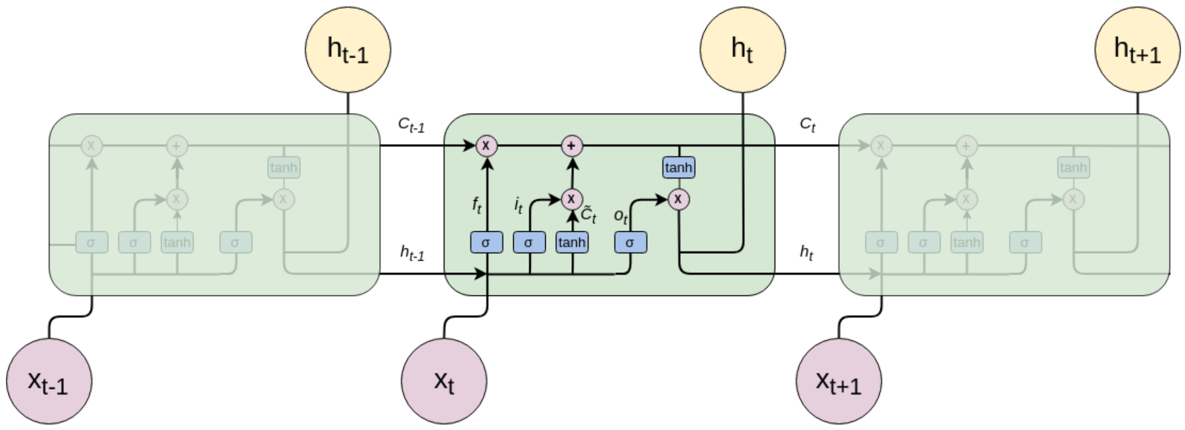


Figure 6: *LSTM network preview*

4 Proposed LSTM model

Just like the RNN, LSTM architecture is also formed as a chain structure. Difference is that every cell has four in-layers each performing specific operations on input data and interacting with others in a special way, general overview of LSTM network is shown on Fig. 6.

An LSTM cell is consisted of 5 components which allow it to model both long-term and short-term data:

- cell state
- hidden state
- input gate
- forget gate
- output gate

Cell state, presented with the upper horizontal line in Fig. 6, runs through the whole cell and is affected by some minor linear changes.

Gates are the mechanism that allows information to be passed in selective fashion from the LSTM to the cell state. They consist of a sigmoid layer, which outputs value between 0 and 1 describing value of each component, and a multiplication operation. There are three types of gates in LSTM network:

- Forget gate - determines which information to remove from the cell state

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- The second gate - decides which new information should be written to the cell state. First, the sigmoid function decides which values to update and then the *tanh* layer creates a vector of candidates to be added to the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$C_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- The output gate - decides what is going to be output of the concrete cell. The output is based on the filtered version of the cell state.

$$o_t = \theta(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

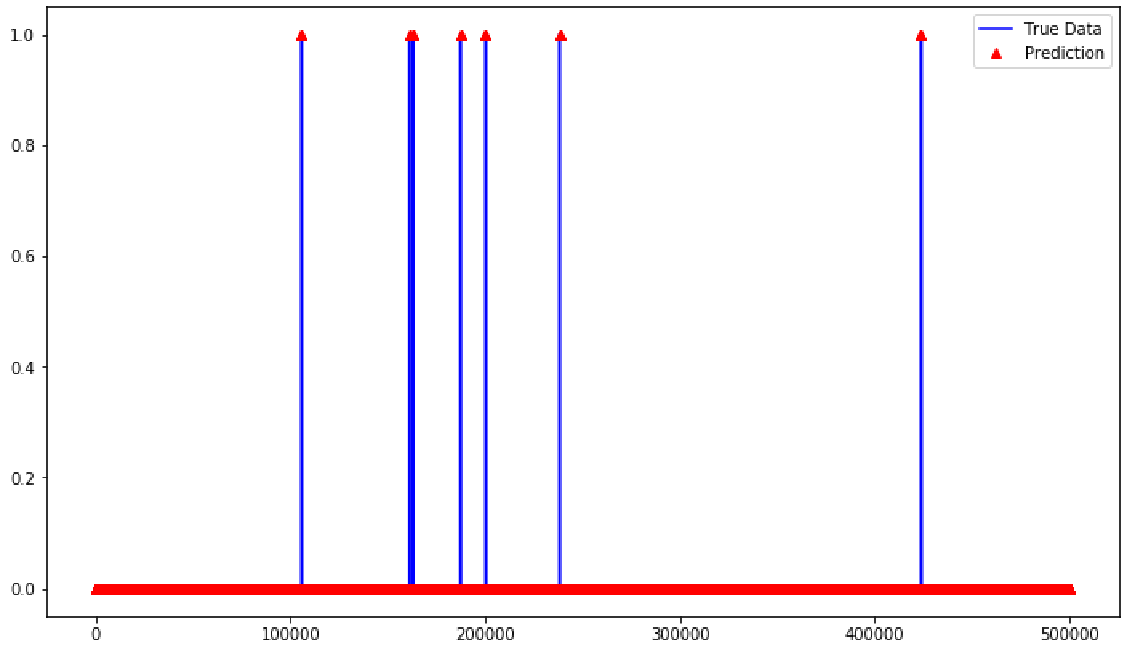


Figure 7: *LSTM output*

In this case, concrete LSTM model for predicting time series data was developed using Keras, open source neural network library written in Python which is capable of running on top of TensorFlow. Model consists of 4 layers where each layer has 50 neurons which structure was described in Fig. 6. The main idea was that the LSTM model successfully predicts when will transmission of observed device happend. After the model had been built, network was fed using data consisting out of sequential time points with joined 0 or 1 depending on the success of transmission of LoRa message for that moment. Before actual load of the data, some preprocessing is required. The way LSTM layers work is by taking in an array of 3 dimensions (N, W, F) where N is the number of training sequences, W is the sequnce length and F is the number of features of each sequence. Sequnce length can be treated as hyperparameter whose value will affect (posibbly improve) overall quality of the model. Sequnce number is, in this case, arbitrary sized window which gives network the ability to detailly observe the data and learn how to build up a pattern of sequences based on the received sequental data. The sequnces are constructed as sliding windows shifted by 1 after every iteration. Number of feature is in this case one since its purpose is just define activity of the specific device.

The model is built by having a potential to predict single time point in the future or to predict a full sequnce (error prone). This overview is covering prediction in point-by-point way.

5 Results

After only one training epoch results were starting to look promising. Training was performed on 80 percent of data, where 20 percent of the data was left for testing and evaluation. On the Fig. 7, blue lines are representing actual device activity while the red triangles are predicting, based on historical events, when the actual events should happend. RMSE score for testing data is 0.1. On the x-axis each point is representing one second in the future where 0 second is presenting current moment, on the y-axis there can only be 1 or 0, for device will be active in the specific second or it won't be active in the specific second in the future.

The core idea of this project is to assess the probability that device will be active for given time period in the future. With results shown on Fig. 7 there is a simple way to do probability estimation, shown on Fig. 8:

$$\mathbb{P}[N_t = k|X_P] = \mathbb{E}[\mathbb{P}[N_t = k|X_s]]$$

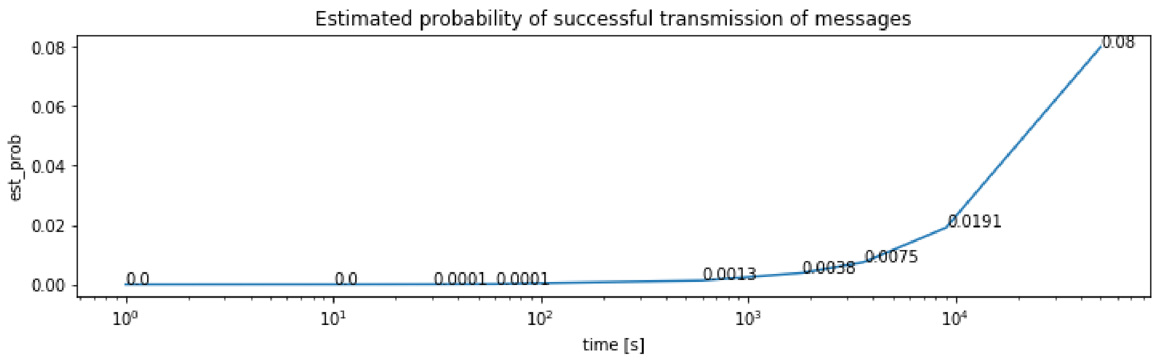


Figure 8: *Estimation of probability of device activation on the test data*

where left side is overall estimated probability considering whole data set and right side is expected value when all subsamples are taken into account.

$$\mathbb{P}[N_t = k|X_s] = \frac{1}{s-t}\sum_{i=0}^{s-t}\mathbb{1}[\sum_{j=0}^{t}X[i+j] = k]$$

Estimation of successful transmission of the message for specific LoRa device for the observed subsample is shown in the equation above.

- First, for the t number of seconds we want to define estimate of probability, we count how many times device was active.
- The same process is applied on the next t seconds in subsample where next t seconds represent time window moved by 1 second in the direction of x-axis.
- When the count for the whole sample is finished, we divide the number of device activations and number of seconds of sample s.
- Lastly, probability estimation (if the time invariance is assumed) is avereged value from samples for the specific window size.
- In addition, final result can be represented as quality of LSTM model times probability estimate: $\mathbb{P}_t = (1 - RMSE) \times \mathbb{P}[N_t = k|X_s]$

As window size (number of seconds for which we want to estimate probability of activation) rises so does the probability of activation. Finally, for the window size equals to sample size, probability should be 1 in most cases. Probability distribution is shown on Fig. 8.

6 Conclusion

TBA

References

[1] Ratnadip Adhikari and R. K. Agrawal. “An Introductory Study on Time Series Modeling and Forecasting”. In: *LAP Lambert Academic Publishing* (Feb. 2013).

[2] Aloÿs Augustin et al. “A Study of LoRa: Long Range and Low Power Networks for the Internet of Things”. In: *Sensors* (Oct. 2016).

[3] J. de Carvalho Silva et al. “LoRaWAN — A low power WAN protocol for Internet of Things: A review and opportunities”. In: (2017), pp. 1–6.

[4] Sepp Hochreiter and Jurgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* (Sept. 1997).

[5] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2 (1989).

[6] Robert Miller. “LoRa Security: Building a secure LoRa solution”. In: *MWR Labs* (Mar. 2016).

[7] Christopher Olah. “Understanding LSTM Networks”. In: *colah’s blog* (Aug. 2015).

[8] Guoqiang Zhang, B.Eddy Patuwo, and Michael Y. Hu. “Forecasting with artificial neural networks: The state of the art”. In: *Elsevier BV* (July 1997).