Probability of Successful Transmission of Uplink Messages in LoRaWAN

Ante Lojić Kapetanović

June 4, 2019

Department of Electronic Systems @AAU

Content

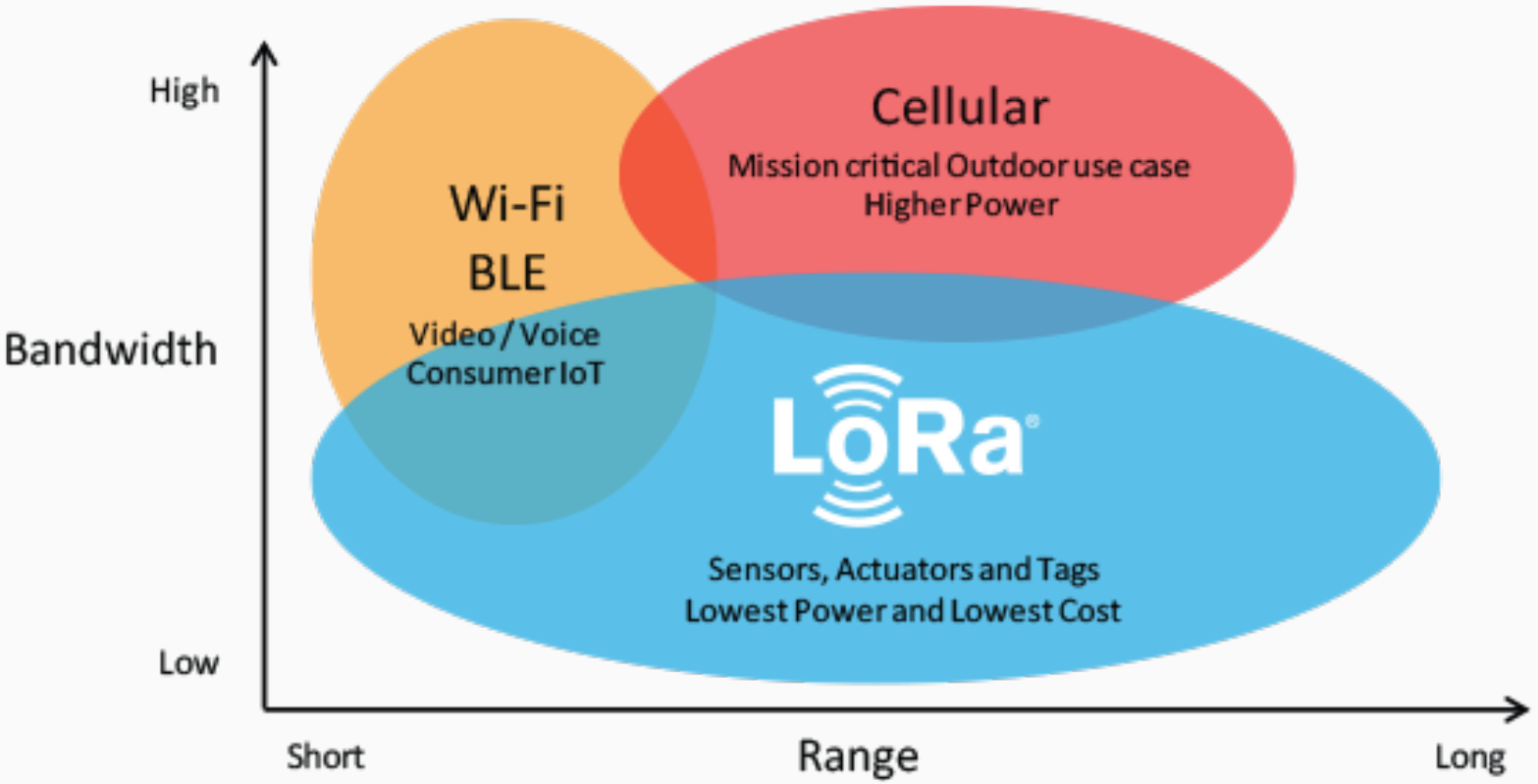
Introduction

LoRaWAN

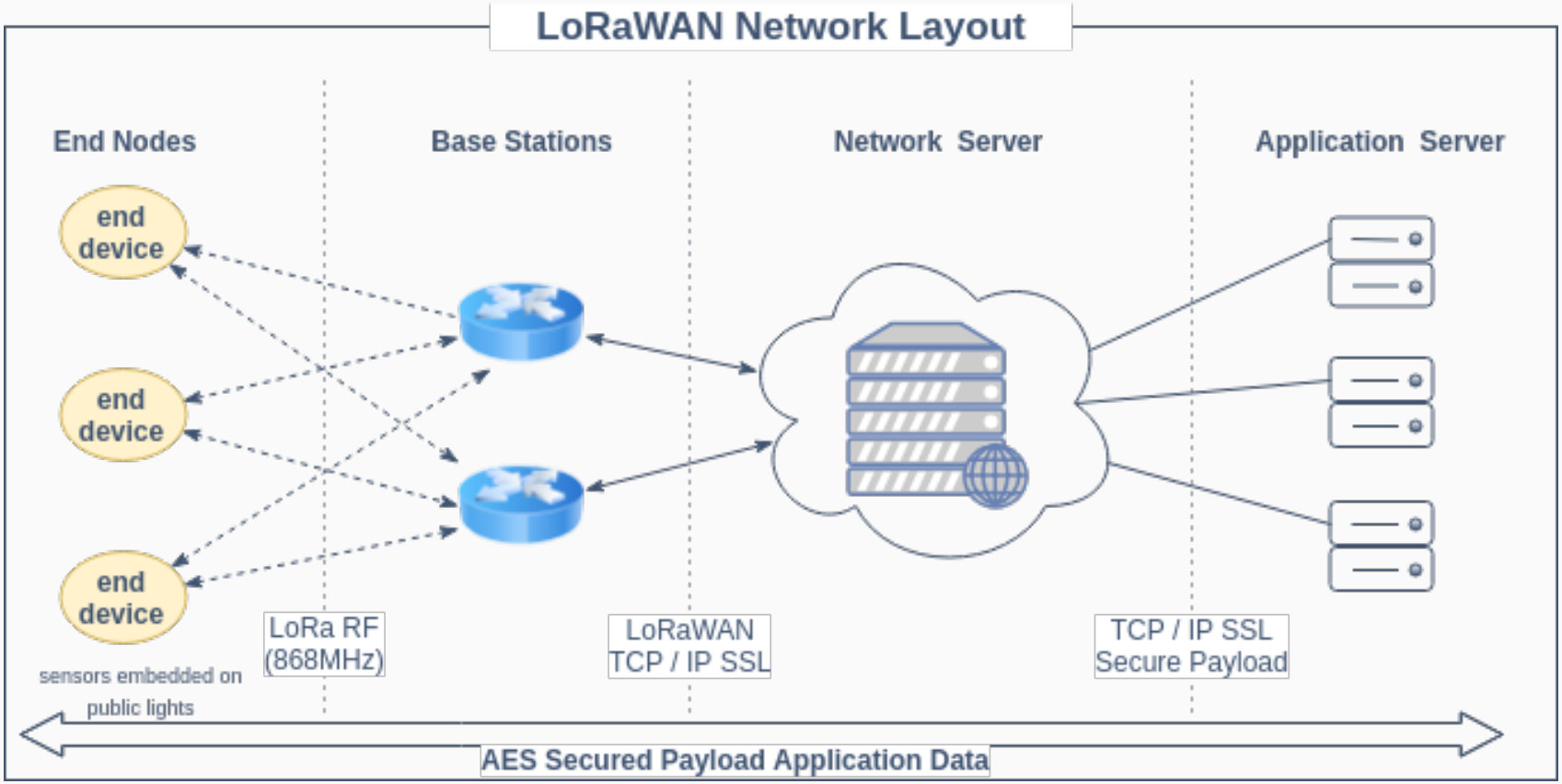
Long Range Wide Area Network - low power, wide area media access protocol built on top of LoRa designed to wirelesslly connect *things* to the Internet.

Perfect fit for network of IoT devices that:

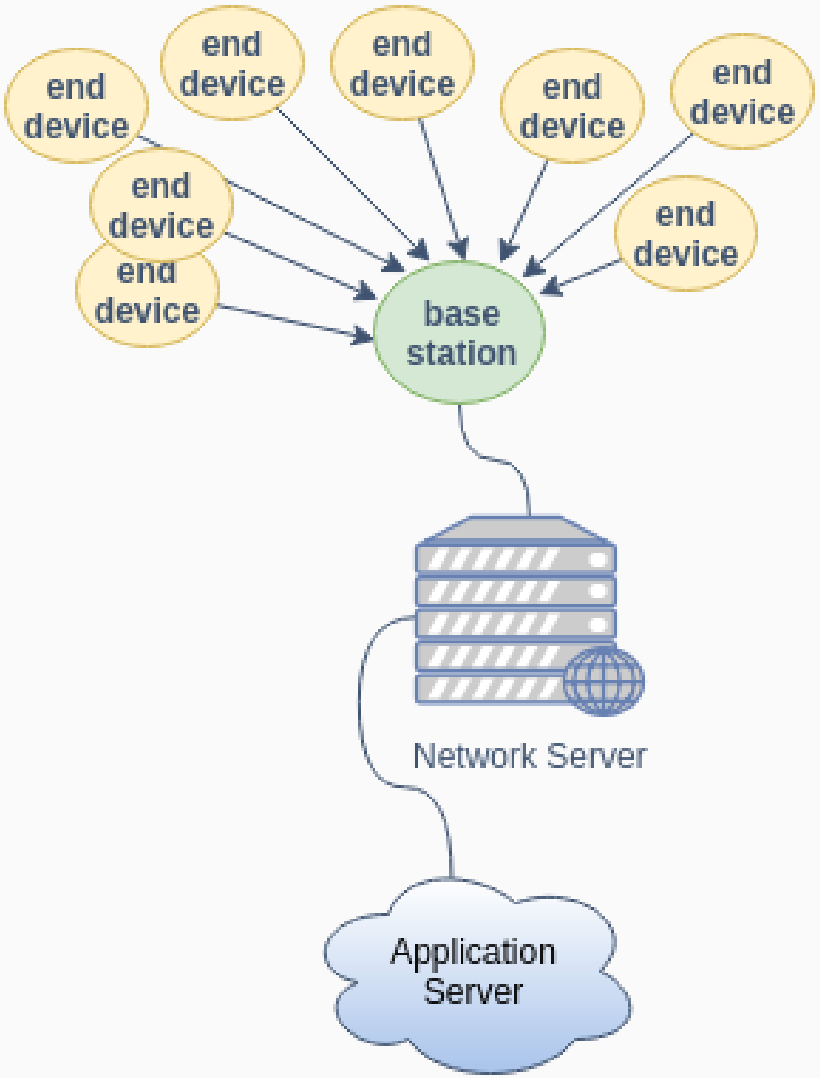
- are not power hungry
- require minimum bandwidth
- send data not to often


High
Bandwidth
Low
Short
Range
Long
Wi-Fi
BLE
Video / Voice
Consumer IoT
Cellular
Mission critical Outdoor use case
Higher Power
LoRa
Sensors, Actuators and Tags
Lowest Power and Lowest Cost

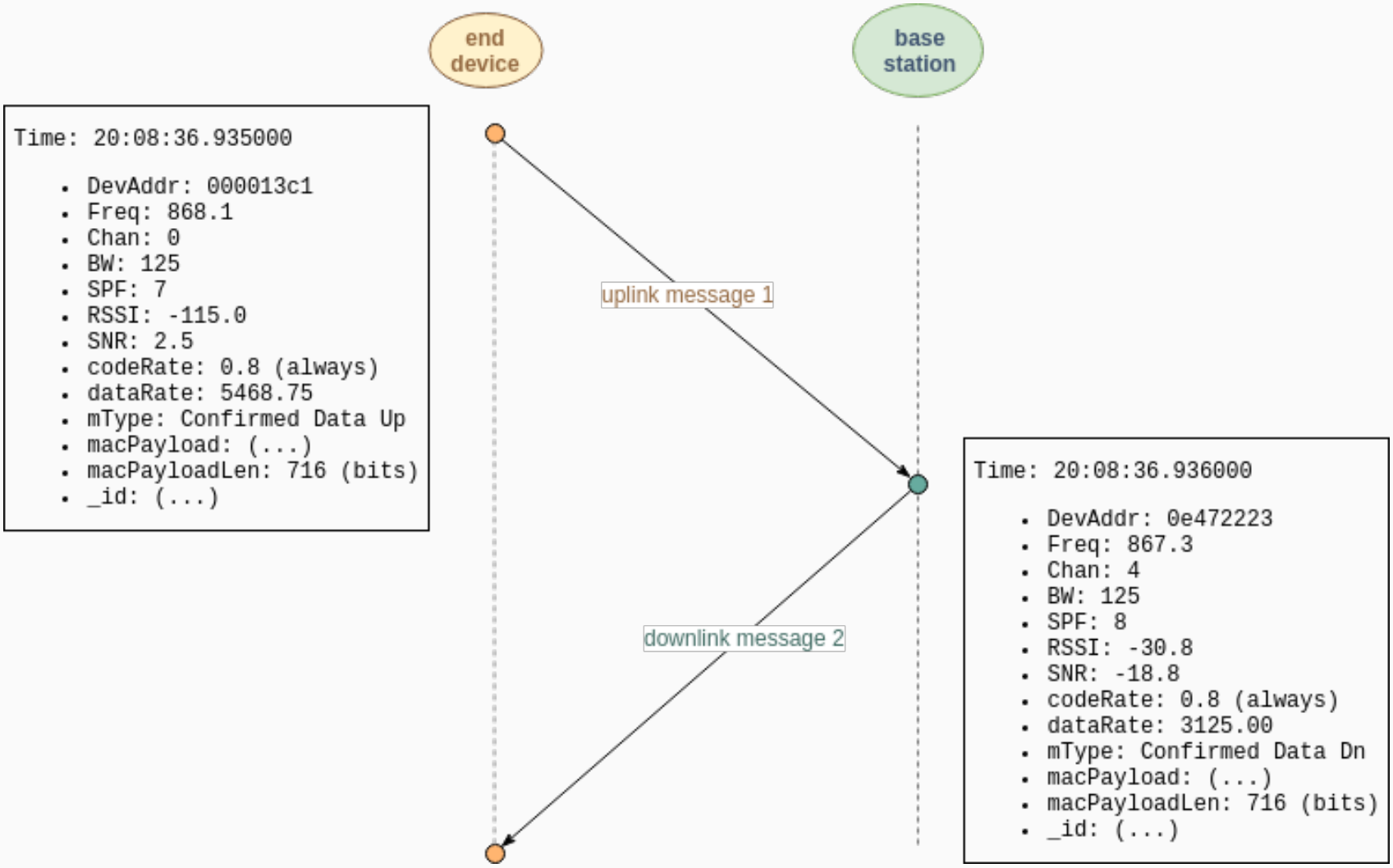
Infrastructure overview



Svebølle topology



Communication model for Svebølle deployment



Project goal

Is to apply forecasting methods on measured data from the single base station in Svebølle and examine whether there is a possibility of predicting successful transmission of LoRa messages for given future time interval.

Predicting future events

Structure of the measurement data

- Data was collected during the period of nearly 5 months
- 689k measurements are captured on a single base station
- Measured features
  - Time
  - DevAddr
  - Freq, Chan, BW, CR, DR
  - RSSI, SNR
  - crcStatus, mType,macPayload

Transformation to time-series

Time-series data is a series of data points indexed in time order and taken at successive equally spaced points in time.

To create time series data from given data set, the only important label is wheter the end device has successfully transmitted the message or not. For each time point, where every time point is observation in seconds there is activity flag assigned:

- 1 - device was active for observed time point
- 0 - device was not active for observed time point

Prediction models

- Moving average / weighted moving average
- Autoregressive integrated moving average (ARIMA)
- Holt's winter method
- Vector auto regression (VAR)
- Recurrent neural network (RNN)
- Reinforcement learning (RL)

Proposed RNN model

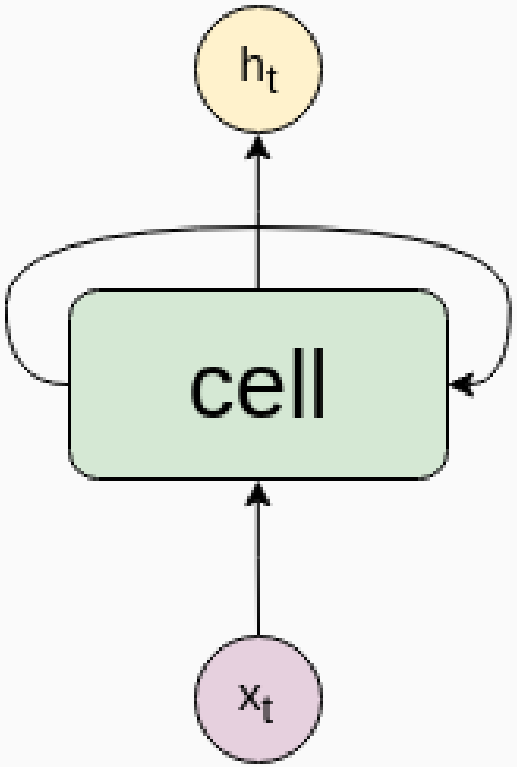
Artifical neural network

Recognizing regularities and patterns of the given data, learning from past experience and providing inference on the output.

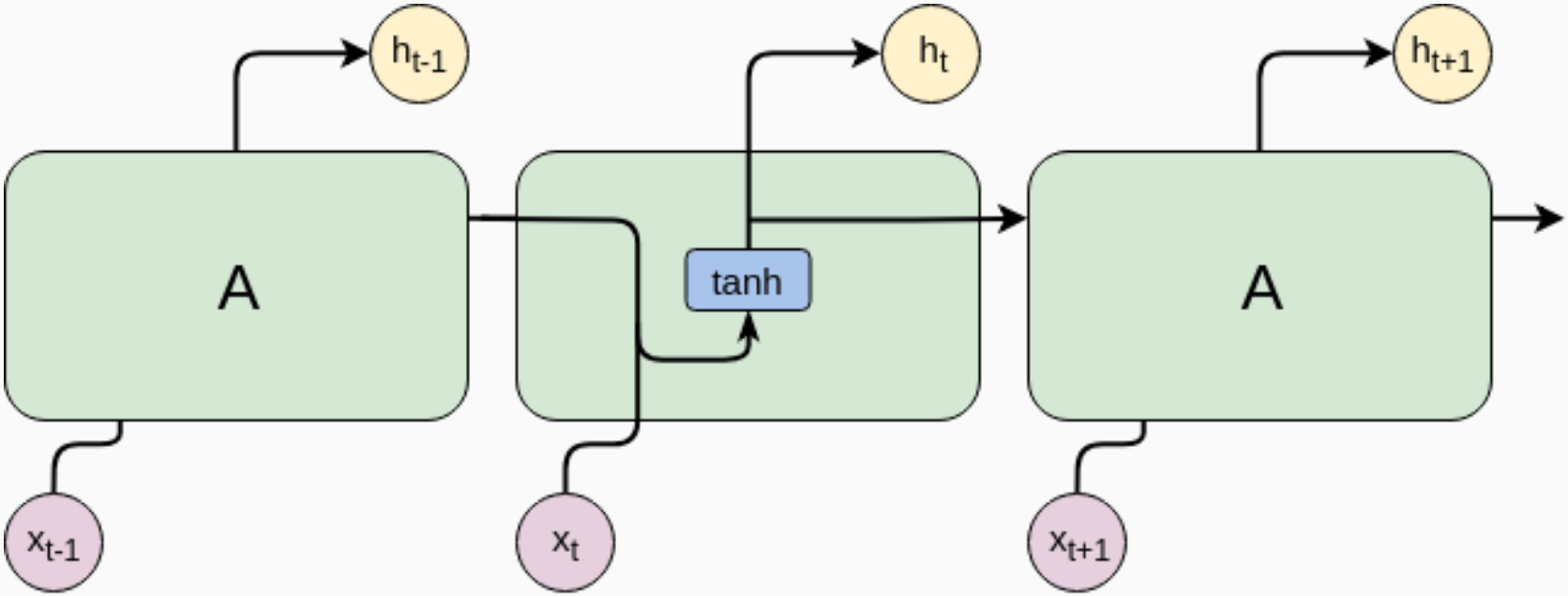
Vanilla ANNs are not perfectly suitable for time-series forecasting because they lack persistance.

Reccurent neural network

RNNs contain loops (feedbacks) allowing information to persist.



Unpacked RNN



Short-term memory problem

If a sequence is long enough, RNN will have a hard time carrying information from earlier time steps to later ones. The reason is that during back propagation, RNNs suffer from the vanishing gradient problem (gradient shrinks during back propagation time).

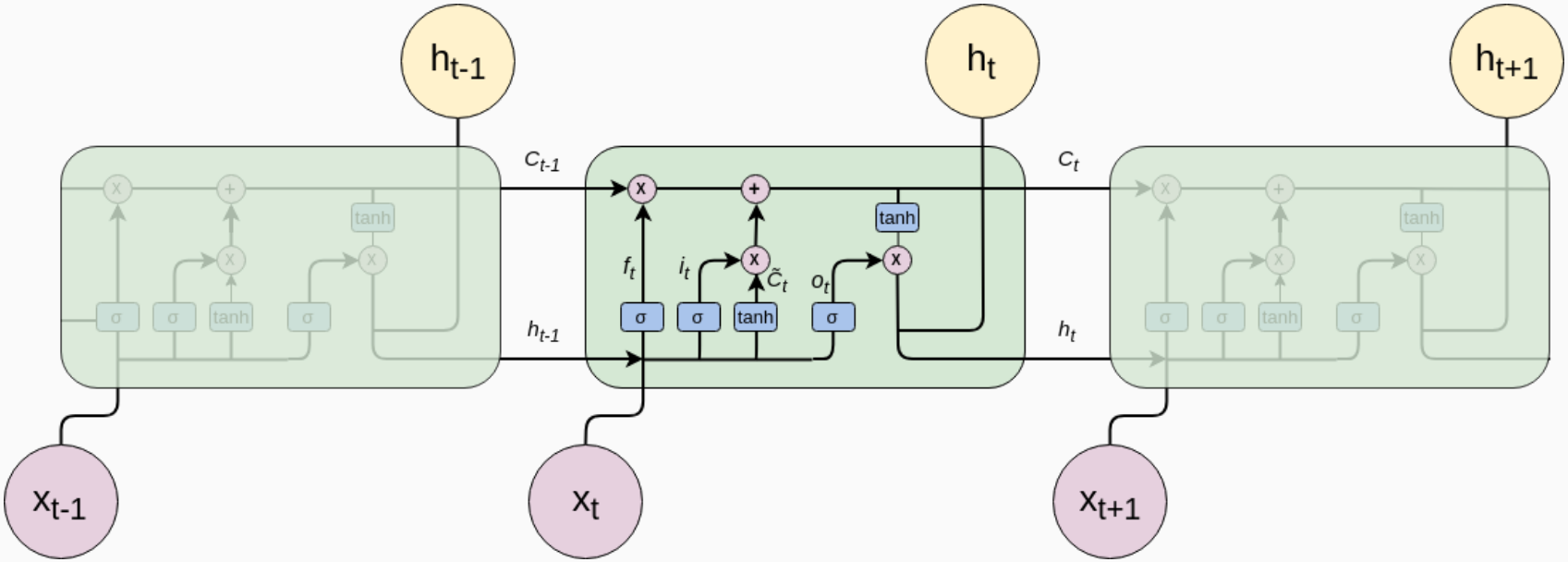
$$\textbf{new weight} = \textbf{weight - learning rate * gradient}$$

$$\text{gradient} = 0 \rightarrow \text{new weight} = \text{weight} \rightarrow \text{RNN stops learning}$$

LSTM networks to the rescue

Changing the inner structure of a cell made possible that the flow of information is regulated.

Inside each LSTM cells there are mechanisms called gates that can learn which data in a sequence is important to keep or throw away.


h_{t-1}
h_t
h_{t+1}
C_{t-1}
C_t
f_t
i_t
$\tilde{C}_t$
o_t
h_{t-1}
h_t
σ
tanh
x_{t-1}
x_t
x_{t+1}

Forget gate

Determines which information to remove from the cell state using sigmoid function that outputs value between 0 (completely reject) and 1 (completely accept)

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

where

x_t is current input vector,

h_{t-1} is previous cell's output value,

W_f is associated weight,

b_f is added bias.

Input gate

Decides what new information should be written to the cell state.

Firstly, the sigmoid function decides which value to update:

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

where

x_t is current input vector,

h_{t-1} is previous cell's output value,

W_i is associated weight,

b_i is added bias.

Cell state

...after that, *tanh* layer creates vector of candidates for the cell state:

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

where

x_t is current input vector,

h_{t-1} is previous cell's output value,

W_C is associated weight,

b_C is added bias.

Output gate

Decides what is going to be output of the concrete cell.

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

where

x_t is current input vector,

h_{t-1} is previous cell's output value,

W_o is associated weight,

b_o is added bias.

The output is based on the filtered version of the cell state.

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

where C_t is the new cell state calclulated as: $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

Model configuration

Configuration

- loss: MSE
- optimizer: ADAM
- layers: 3 LSTM (CuDNN implementation)
- droput layer with rate of 0.2 after each LSTM layer
- batch normalization layer after first two dropouts
- last dense layer with only 1 layer has ReLu activation

Probability of device activation for future time period

Preprocessing

1. cleansing
2. clean structured data $\rightarrow$ time-series data
3. creating sequences
4. randomizing sequnces
5. train-validation split

Training

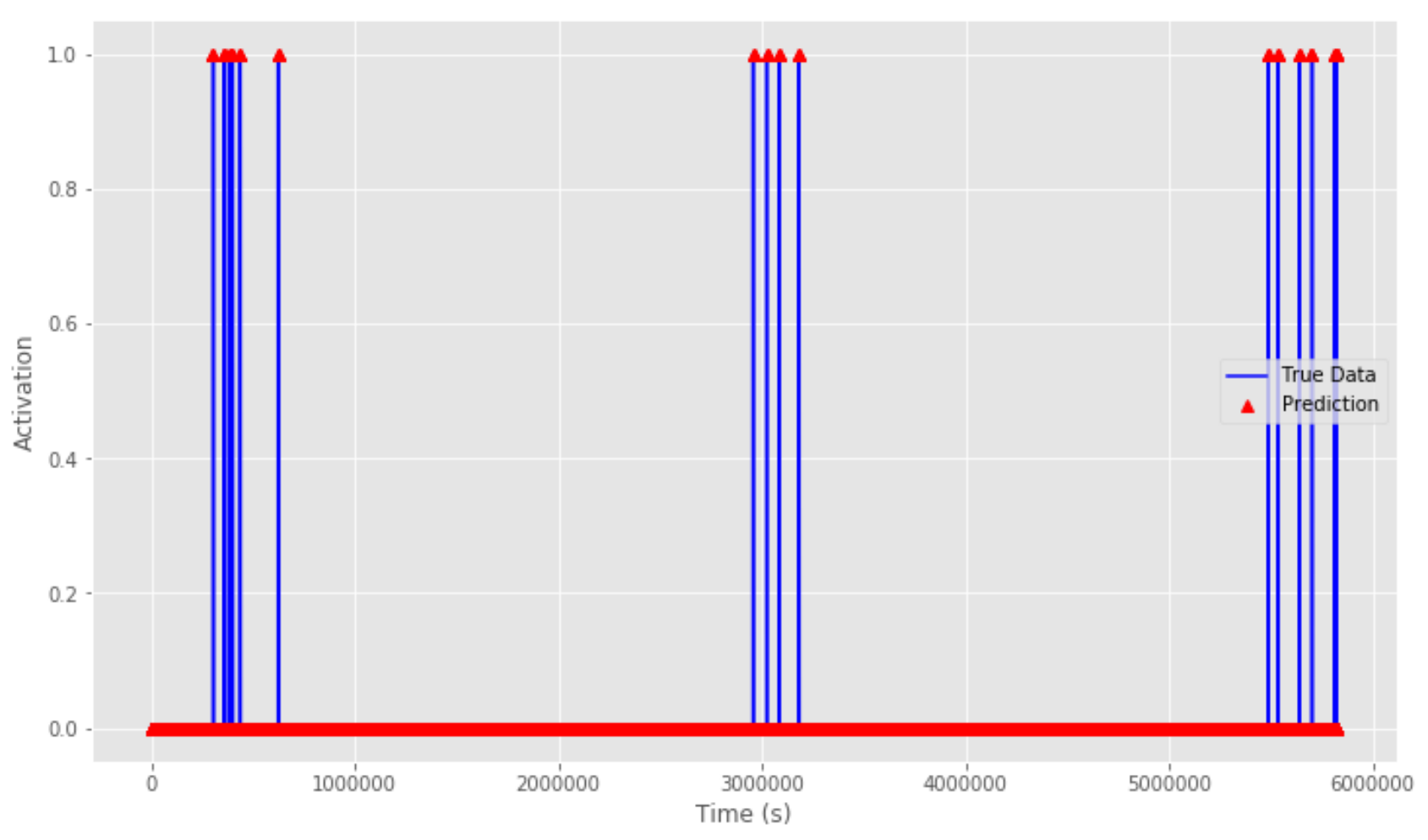
In-memory training:

- 10 epochs
- 64 batch size

using

```
callbacks = [
    EarlyStopping(...),
    ModelCheckpoint(...),
    TensorBoard(...)
    ]
```

Validation results



Probability of successful transmission

For given period of time (**n** seconds) probability of successful transimission of LoRa uplink message should rise up as time period gets bigger (**n** gets bigger).

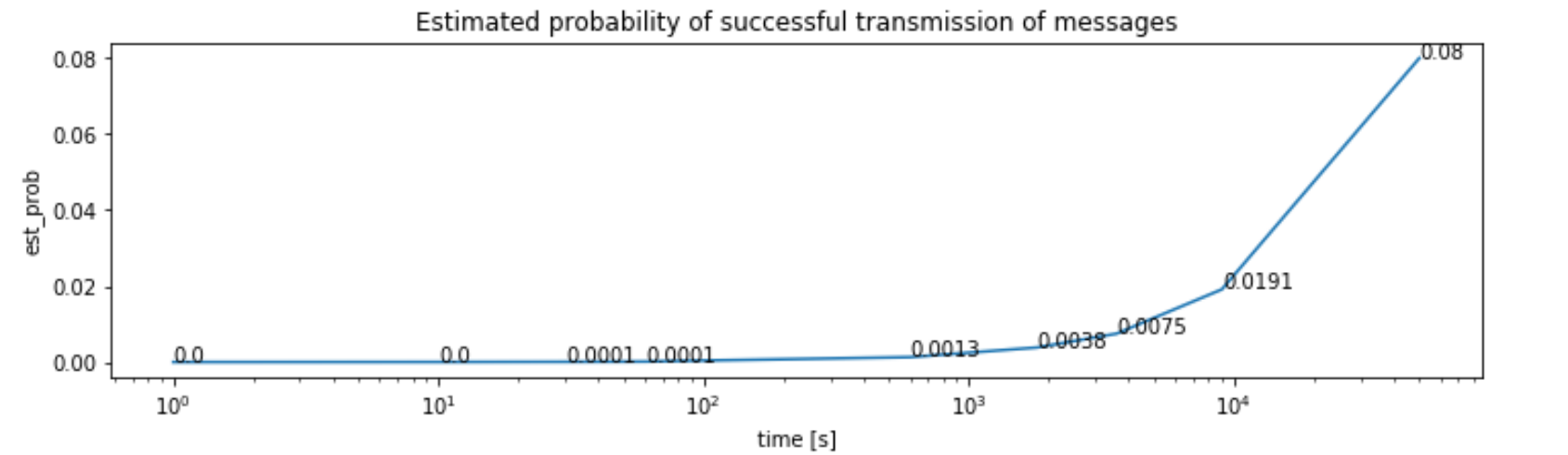
$$n = 0 \Rightarrow p_{st} = 0$$

$$n \in < 0, +\infty > \Rightarrow p_{st} \in < 0, 1 >$$

$$n = \infty \Rightarrow p_{st} = 1$$

Probability can be calclulated as:

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


Estimated probability of successful transmission of messages
est_prob
time [s]
0.08
0.06
0.04
0.02
0.00
10^0
10^1
10^2
10^3
10^4
0.0
0.0
0.0001
0.0001
0.0013
0.0038
0.0075
0.0191
0.08

Conclusion

Future work

- **Multivariate time-series data** instead of univariate time-series data: observe not only previous historical events for specific device but also activation of other devices
- **Longer training sequences**
- Future **sequence predictions** instead of point-to-point predictions

Questions?