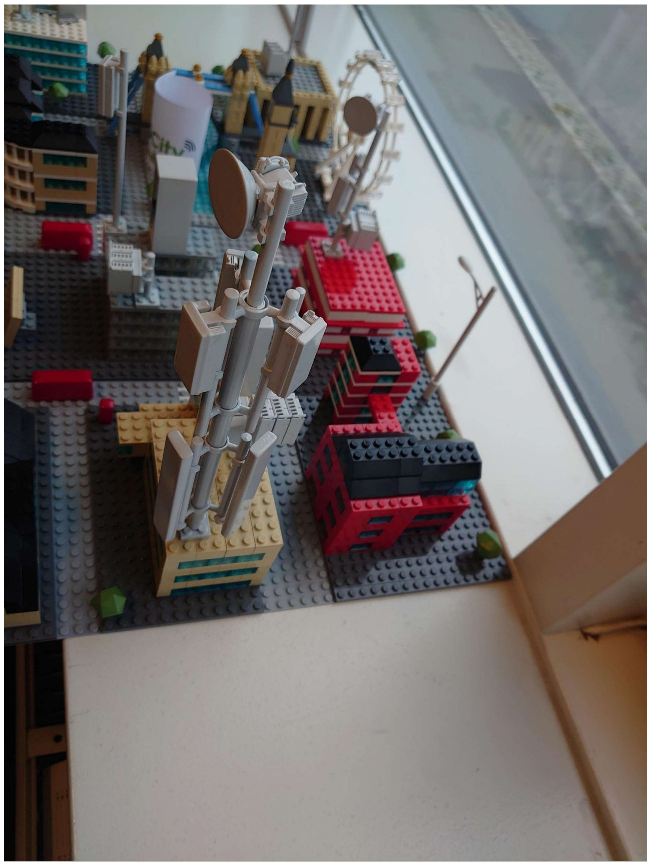


AI CLASSIFIER FOR E-BAND RADIO LINKS

How my Prototype was built

ABSTRACT

This is how you could build an AI prototype that classify outage and events on wireless links.

Glenn Landgren

Transmission Network Engineering Manager

CONTENTS

AI classifier for E-Band radio links

This is how you can build an AI prototype that classify errors and events on wireless links.

1. INTRODUCTION

Today, pro-active maintenance and performance monitoring is mainly carried out by using hardcoded thresholds. This is good for many problems such as finding misaligned antennas and LOS problems by comparing planned vs. actual received signal level. With some coding, you can also use thresholds to find links stuck on low modulation.

For other problems, such as finding unstable antennas sensitive to wind, or to understand when errors and unavailability is caused by wet snow, then pattern recognition seem to be required.

The purpose with this document is to share a way to do pattern recognition on point-to-point E-band radio links by using machine learning (AI).

You will see a multi-class classifier where an algorithm is trained to learn to recognize patterns for each event specified. (E.g. Errors caused by - unstable antenna, rain, wet snow, HW problem, etc.)

I will go through everything required to create an end to end prototype. From creating the machine learning model (this article), to a backend App in Python Flask (next article).

With the attached code, in Jupyter notebooks, you will see how the algorithm was built and trained.

I have focused on E-band radio links because this is the band that will be used for wireless 5G backhaul. The E-band operate at 70-80GHz and support 10Gbps on a single carrier.

2. CONTEXT

2.1. Background

E-Band Radio Links provide fiber like capacity and will play an important role in 5G roll out. BUT, with wireless links – compared to fiber – there are a lot of more different factors that may cause errors, outage and unavailability. It is important to understand how, when and why those different events occur, in order to limit consequences or prevent them from happening again.

Typical events that cause errors, unavailability or limited throughput are:

- Unstable antenna, e.g. the antenna or antenna pole is moving/ twisting.
- Wet snow stuck on antenna
- Rain
- Different kind of Line of Sight problems
- Faulty equipment

Operators challenge is:

- To get early alerts on bad performance
- To understand the root cause of errors and outage
- To do preventive maintenance that solve issues before they become bigger problems
- To develop tools to carry out and maintain fact-based performance comparisons between fiber and wireless transport media

The last bullet is of high importance and I believe there is too much guessing today.

2.2. Problem description

For wireless transport to become successful in the era of 5G, performance monitoring, event classification and root cause analyses needs to be automated.

Question is: Can we use our existing performance data, to train machine learning algorithms, to do accurate event classification?

The Outage/ Events we will classify are:

- Unstable antennas
- Wet snow
- Rain
- Equipment (HW) fault
- Hardware
- *And many more*

We want to understand what events that cause most Errors and Outage. When and Where.

3. SYSTEM DESCRIPTION AI PROTOTYPE

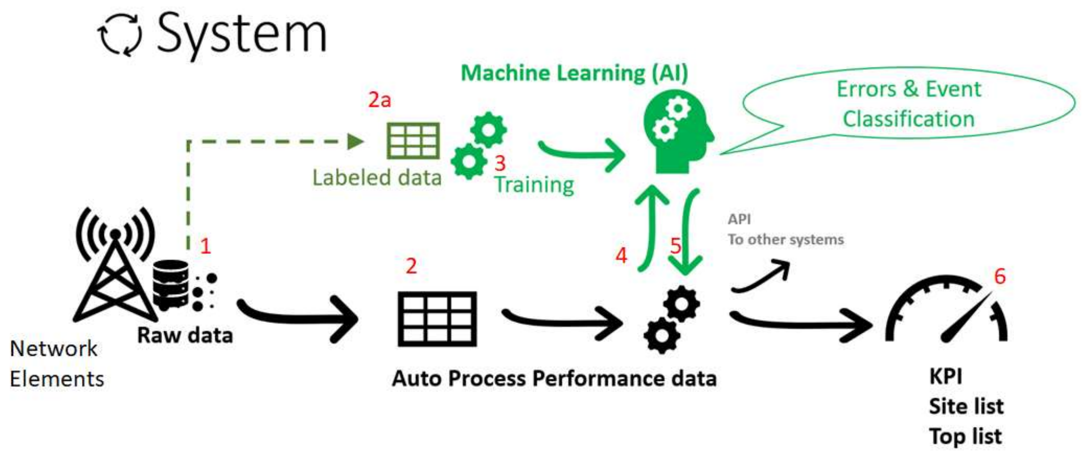
3.1. Overview

Performance Data is collected by an element manager. For each radio terminal (network element) data is collected every 15 minutes.

The element manager saves and store the data in .CSV format. It is large files that contain 15-minute data for all network elements. There is one file per type of performance counter. E.g. adaptive modulation.

The system overview is outlined below. Please refer to system schematic below.

1) Collected data from network elements. Saved and stored as CSV files.
2) Data is auto processed, and data frames are combined.
   a) The same script creates data frames that are used to label train examples.
3) Training of machine learning algorithm
4) Real network data is sent through the trained machine learning model
5) The machine learning model return an event classification for each network element, both on 15-minute and 24-hour level.
6) The classification result is presented together with other performance data on top-lists, KPI graphs etc.

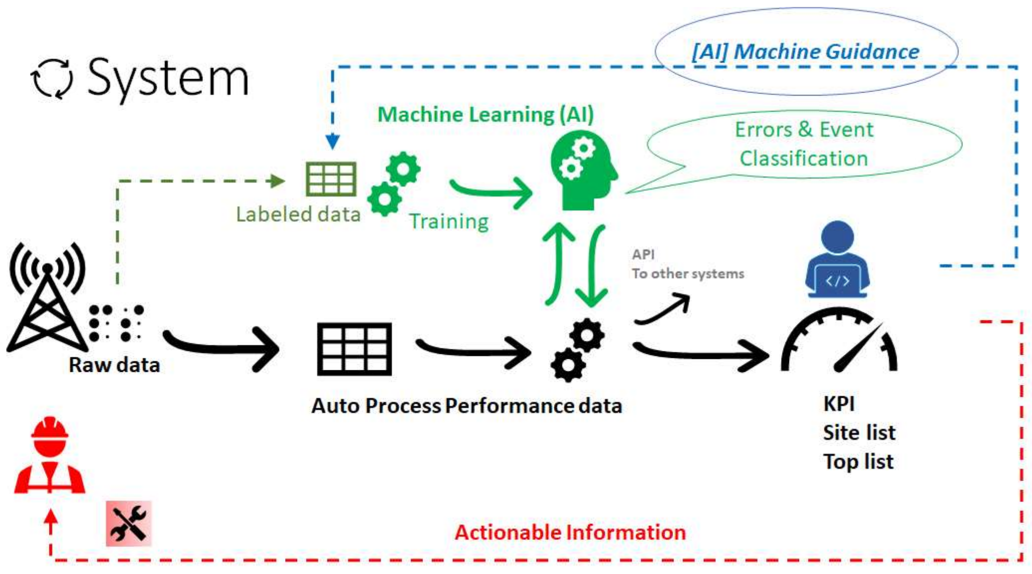


This system is in use at Hi3G today for testing purposes. It means that all E-band MW links are classified daily by two Algorithms. One on 15-minute level and one on 24-hour level. The predicted class is presented for each radio terminal together with other pm data such as actual errors and bandwidth utilization. Both Algorithms will be presented in detail later in this document.

3.2. Machine Guidance

The system includes something I call "Machine Guidance". In short it means that when an engineer finds a fault, it can be classified with one click (after selecting element id and time). By doing this well, we implement a learning loop that could potentially develop a powerful Algorithm.

The long-term goal is to automatically generate actionable information, that goes straight to the field engineers. See schematic below.



4. COLLECTED TRAINING DATA

4.1. Dataset overview

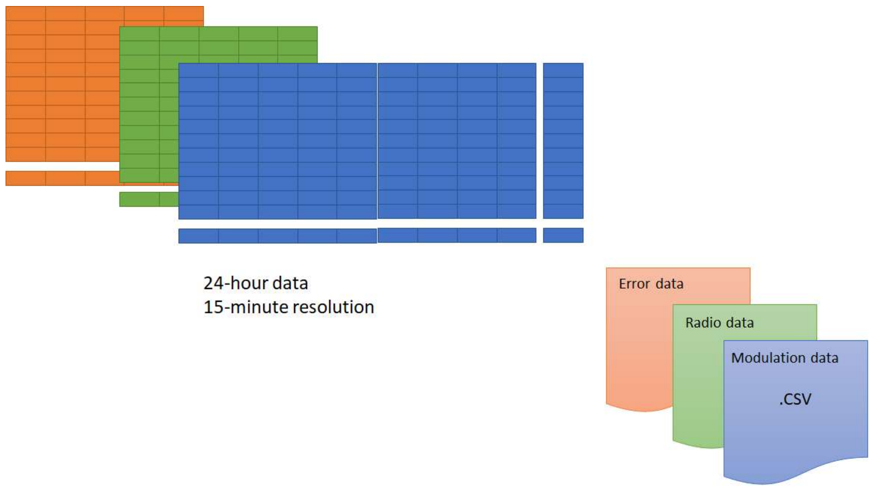
The dataset is used to train Machine Learning algorithms. Later in this document the selected machine learning models are explained in detail.

The **dataset** combines three different data files. Each file contains all network elements with a specific type of data. Each file contains data for 24-hours, on 15-minute resolution.

Topology information is fetched from planning tool to determine the other side (far-end side) of Radio link

The three data files we will merge into *the* **dataset** are:

- Received level on radio signal (both near end and far end of the radio link)
- Errors and availability counters (both near end and far end of the radio link)
- Adaptive modulation information (both near end and far end of the radio link)

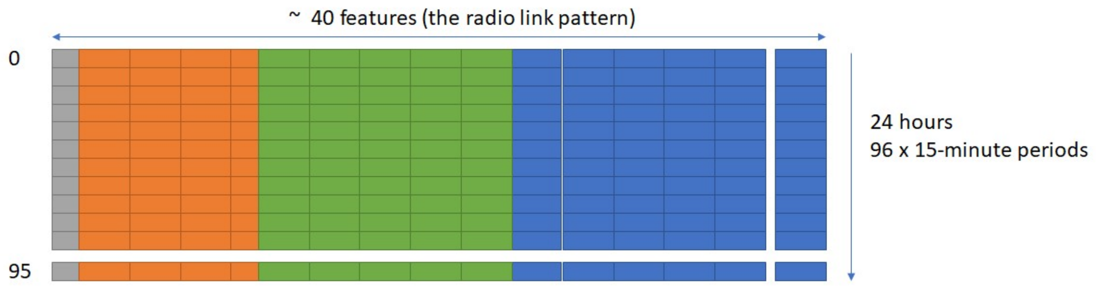


When the dataset is ready (see below) it will have the dimension: m x 96 x ~ 40 features.

- m is the total number of examples (network elements) in dataset
- 96 is the number of time steps for a 24-hour period
- The exact number of features will differ but stays around 40

Below is the structure of the data frame for one radio terminal, 24 hours.

One Radio terminal (Network element) 24h Dataframe



Note! Merging data-frames is done with Python Pandas. It requires some work to merge and align data-frames, but good news is you only do it once! With the "merging script" you will both generate new train examples as well as preparing actual "production data" that will be run through the Machine Learning Model. (The model you will save after you are happy with training your algorithm, - show you later)

4.2. Dataset Details

Shape:

```
Shape of dataset is (57504, 50)
```

Columns:

```
Index(['NeAlias', 'HalfBPSK_Strong', 'HalfBPSK', 'HalfBPSK_Light',
       'BPSK_Strong', 'BPSK', 'BPSK_Light', 'QAM4_Strong', 'QAM4',
       'QAM4_Light', 'QAM16_Strong', 'QAM16', 'QAM16_Light', 'QAM32_Strong',
       'QAM32', 'QAM32_Light', 'QAM64_Strong', 'QAM64', 'QAM64_Light',
       'QAM128_Strong', 'QAM128', 'QAM128_Light', 'QAM256_Strong', 'QAM256',
       'QAM256_Light', 'QAM512_Strong', 'QAM512', 'QAM512_Light',
       'ModChanges_x', 'ES_x', 'SES_x', 'UAS_x', 'ActualReceived_Level_x',
       'ES_y', 'SES_y', 'UAS_y', 'ActualReceived_Level_y', 'ModChanges_y',
       'fading_metric', 'sum_errors', 'hw_fault', 'rx_levels_metric',
       'ModChanges_metric', 'errors_metric', 'UAS_metric',
       'fading_vs_rx_metric', 'Y_15min', 'Y_15min_text', 'Y_24h',
       'Y_24h_text'],
      dtype='object')
```

Original collected features:

- NeAlias is the radio terminal id (network element)
- Modulation types show which modulation the radio is running on (seconds per 15 min period)
- Number of modulation changes per time step
- Errors and Unavailability (ES, SES, UAS)
- Received level (RF signal, dBm)

Suffix '_x' refer to near-end radio terminal
Suffix '_y' refer to the far-end radio terminal (data /features from other side of link)

Below are engineered features. Those features represent a combination of the original features collected from radio terminal.

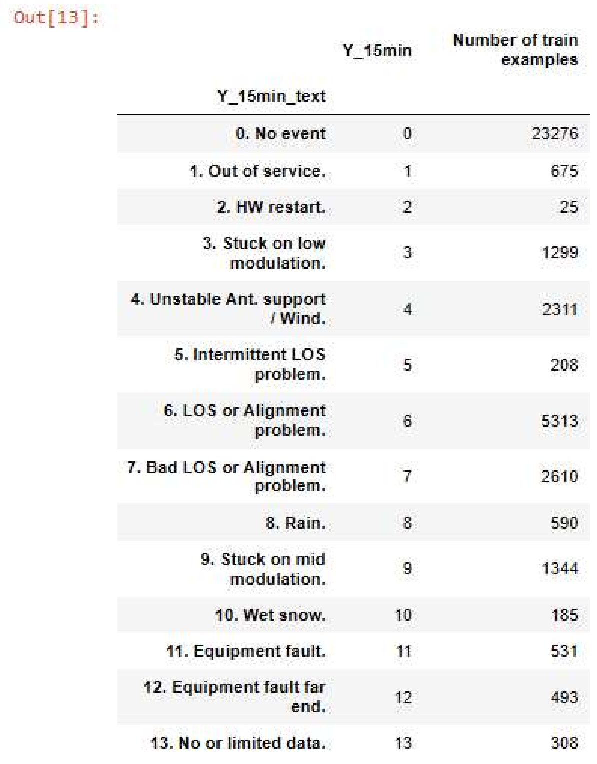
- fading_metric, low modulation gives a high value
- sum_errors, augmenting errors features (ES, SES)
- hw_fault, flag - errors but no modulation changes
- rx_levels_metric, augmenting Received level
- ModChanges_metric, augmenting Modulation Changes
- errors_metric, augmenting errors features again (ES, SES)
- fading_vs_rx_metric, augmenting relationship between fading metric and Rx level.

There are two labels.

- Y_15min, the event class label for each 15-minute period
- Y_24h, the event class label for a 24-hour period. (96x15 min periods)

4.2.1. Class distribution 15_minute label

Each training example has 4 time-steps. (4 x 15-min periods)

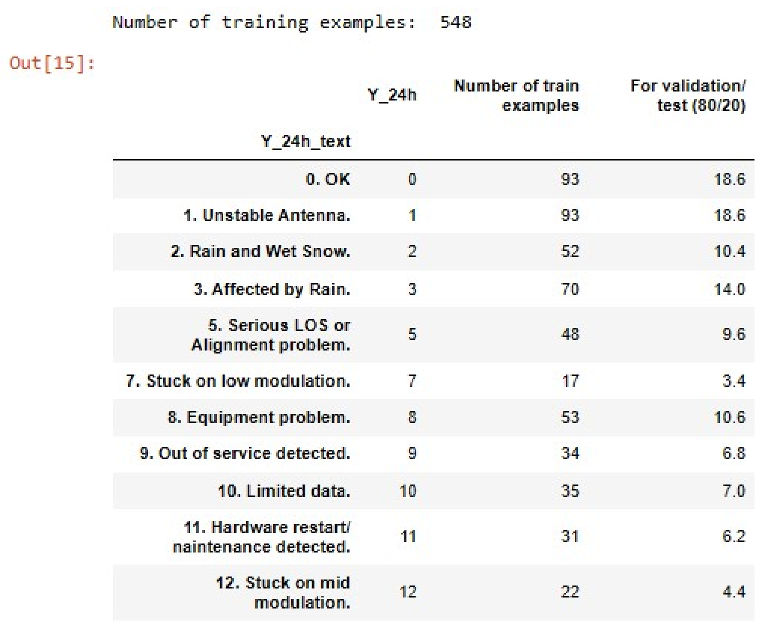


Out[13]:

Y_15min_text	Y_15min	Number of train examples
0. No event	0	23276
1. Out of service.	1	675
2. HW restart.	2	25
3. Stuck on low modulation.	3	1299
4. Unstable Ant. support / Wind.	4	2311
5. Intermittent LOS problem.	5	208
6. LOS or Alignment problem.	6	5313
7. Bad LOS or Alignment problem.	7	2610
8. Rain.	8	590
9. Stuck on mid modulation.	9	1344
10. Wet snow.	10	185
11. Equipment fault.	11	531
12. Equipment fault far end.	12	493
13. No or limited data.	13	308

4.2.2. Class distribution 24_hour label

Each training example has 96 time-steps. (96 x 15-min periods)



```
Number of training examples:  548
Out[15]:
```

Y_24h_text	Y_24h	Number of train examples	For validation/ test (80/20)
0. OK	0	93	18.6
1. Unstable Antenna.	1	93	18.6
2. Rain and Wet Snow.	2	52	10.4
3. Affected by Rain.	3	70	14.0
5. Serious LOS or Alignment problem.	5	48	9.6
7. Stuck on low modulation.	7	17	3.4
8. Equipment problem.	8	53	10.6
9. Out of service detected.	9	34	6.8
10. Limited data.	10	35	7.0
11. Hardware restart/ naintenance detected.	11	31	6.2
12. Stuck on mid modulation.	12	22	4.4

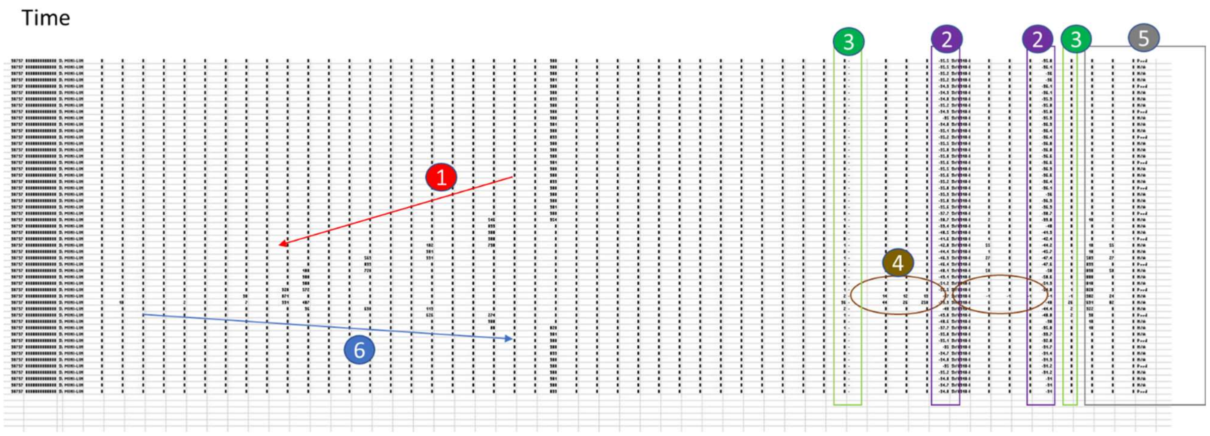
5. PATTERN RECOGNITION

What do we want the machine to learn ?

When you have worked with Wireless transport and Microwave engineering for a long period of time, you get a good feeling for what causes errors and outage just by looking at the data frame with the radio pattern. By letting experts label those patterns we can train a machine learning algorithm to classify the events automatically - anytime. It's a lot of work in the beginning since we need a lot of training examples. Just like when you train an algorithm to understand the difference between cats and dogs.

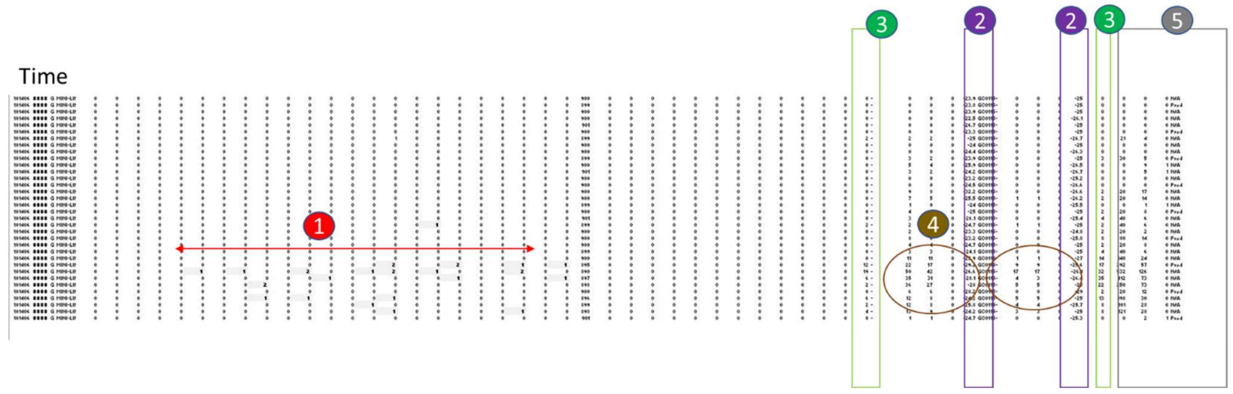
In picture below you see a pattern example of wet snow getting stuck on an antenna. Remember each row is 15 minutes.

1) Modulation decrease slowly as wet snow get stuck on the antenna
2) Receved signal level affected
3) Number of modulation changes
4) Errors and unavailability
5) Other engineerd features affected
6) Modulation increase as snow melt/ fall of antenna



In next picture you see a pattern example of on an unstable antenna affected by wind.

1) Modulation type is scattered along affected time steps.
2) Receved signal level affected, but not like snow and rain
3) Number of modulation changes are high
4) Errors and outage
5) Other engineerd features affected



Those are just examples and there are of course variations in patterns within the same class. But the thesis is that a well-trained algorithm will be able to distinguish between the patterns in the different classes.

6. MACHINE LEARNING MODELS

The goal is to develop an optimal machine learning model that learn to understand the radio signal pattern (the data frame / matrix) with high accuracy.

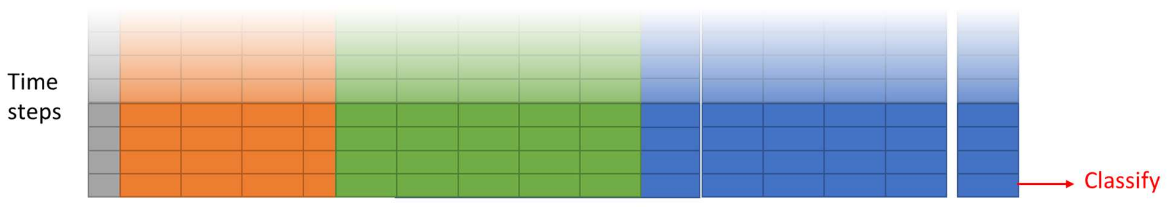
Three types of Neural Networks are presented. Models are explained further below, and all coding can be found in Jupyter Notebooks. Models are built in Python and Tensorflow with Keras API.

1) Fully Connected Neural Network
   - ➢ Used to classify 15-min periods (option 1)
2) Convolutional Neural Network
   - ➢ Used to classify 15-min periods (option 2)
3) Recurrent Neural Network, LSTM many to one
   - ➢ Used to classify 24-hour periods *

Please note that all models all based on supervised learning and require a labeled dataset.

6.1. Classify 15-minute periods

For each 15-minute period we classify, - we will use *n* number of previous 15-min periods.



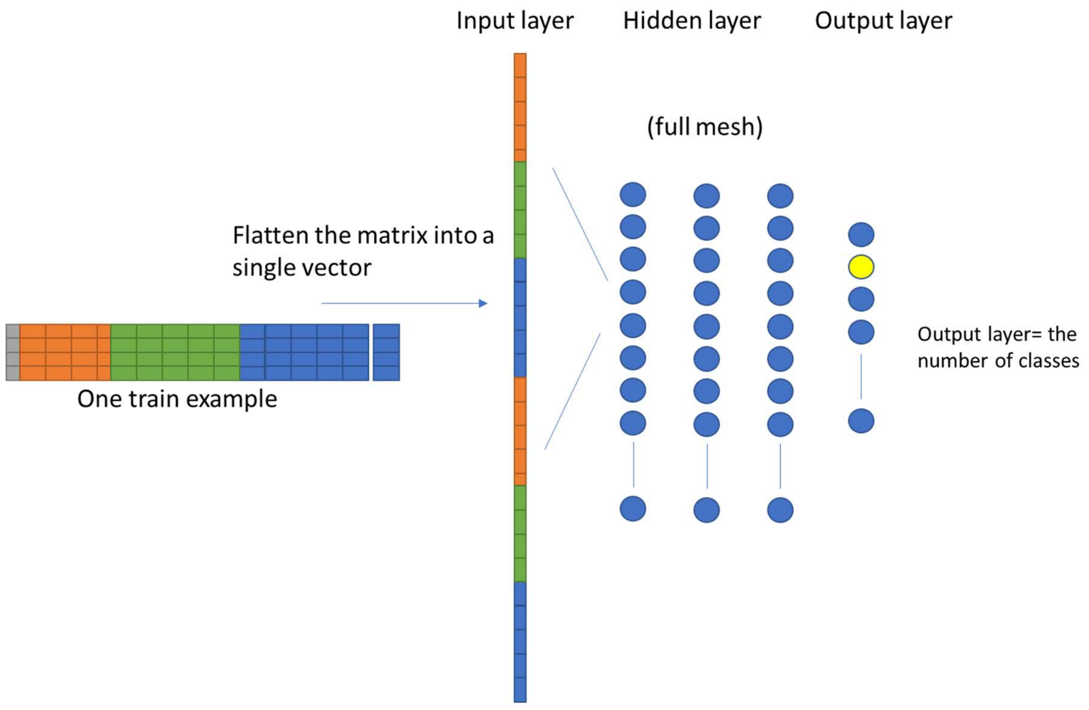
The models shared below is based on 4 rows. It means that, for each 15-min period to classify, we will use the data from the three previous time steps.

Note: When you look at the attached code in Jupyter Notebooks, you will see that its easy to change from e.g. 4 to 8 rows. It would mean that you would add more previous timesteps to classify current timestep.

6.1.1. Fully connected Neural Network, using 4 x timesteps

This is the traditional neural network architecture. The training example (current timestep plus 3x prev. timesteps) will be flattened before it enters the neural network. The number of hidden layers and the number of neurons in each layer can be found (and modified) in attached notebook.

The size of output layer is the same as the number of classes we predict. The neuron that gets activated correspond to the predicted class.


Input layer
Hidden layer
Output layer
(full mesh)
Flatten the matrix into a single vector
One train example
Output layer= the number of classes

6.1.2. Convolutional Neural Network, using 4 x timesteps

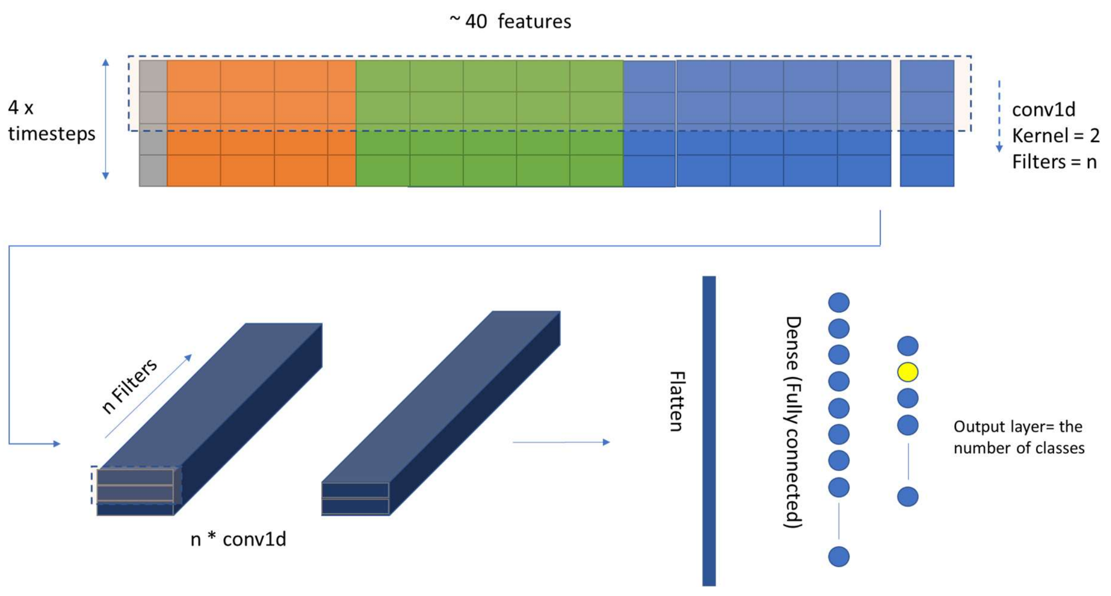
Convolutional neural networks normally work better on time series than the traditional neural network architecture described previously.

For time series classification we use 1D filters, meaning that the filters move along timesteps in one dimension. The filter parameters are multiplied with the features as the filter slide down over the matrix. All parameters are trainable.

In our model we do many convolutions as the many filters slide over the data frame of time * features.

After three conv1d layers the output is flattened and connected is one fully connected layer that connects to the output layer.

The size of output layer is the same as the number of classes we predict. The neuron that gets activated correspond to the predicted class. See picture below.



6.2. Classify 24-hour periods

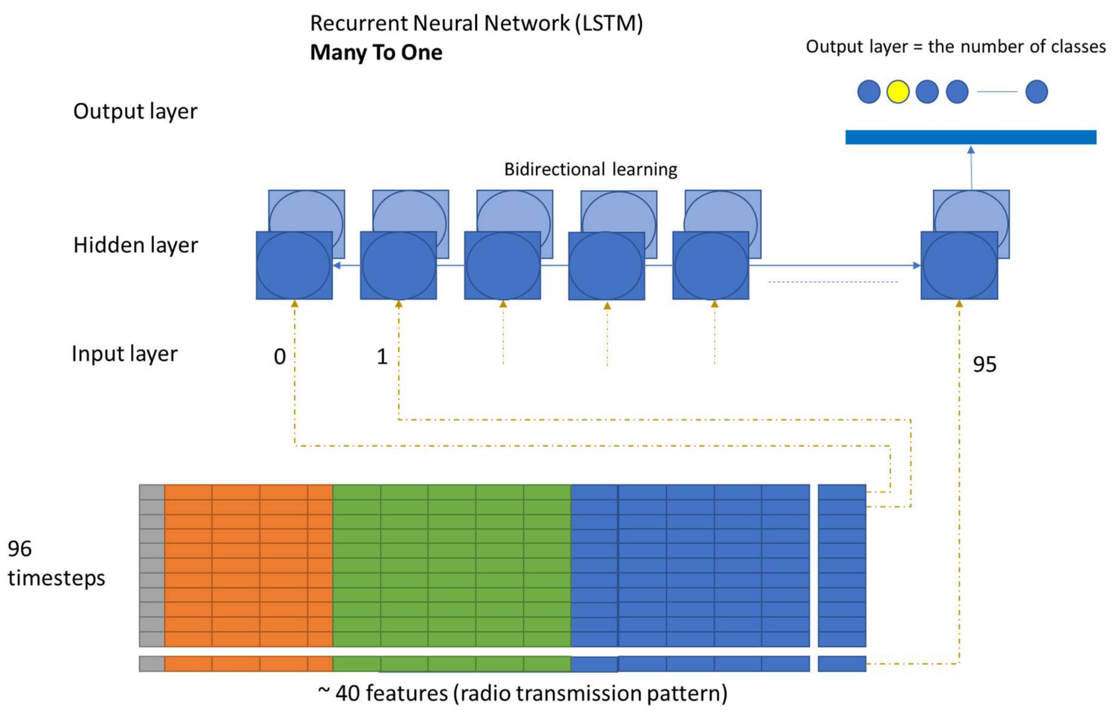
Here we work on a large data frame with 96-time steps. We want to understand and generalize how the radio terminal performed during a 24-hour period, e.g. yesterday. Was it ok? If there were errors or outage, – what was the main reason?

To do this I selected a Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) that "remember" previous time steps (15-min periods). I also use the bi-directional variant, which means that the RNN will learn both from the past and from the future.

The architecture is many-to-one which means that output class is predicted at last time step. I.e. one prediction for the 24-hour period.

Dropout is used at two levels to prevent the model from overfitting to the training set. The size of output layer is the same as the number of classes we predict. The neuron that gets activated correspond to the predicted class.

See model in picture below.


Recurrent Neural Network (LSTM)
Many To One
Output layer = the number of classes
Output layer
Bidirectional learning
Hidden layer
Input layer
0
1
95
96 timesteps
~ 40 features (radio transmission pattern)

7. RESULTS

The **dataset** is divided between **training** set and **test** set.

- Training set is used to train the algorithm.
    - 80% of the data
- Test set is used to *validate* the algorithm on data it has *not* seen during training.
    - 20% of the data

Note that each training example has a true label which tells what class it belongs to.

Results are presented on Test set. The model is trained to get as high **Accuracy** as possible on test set. For validation you will see a **Confusion matrix** where you will be able to compare True Label and Predicted Label for each class predicted. By using a confusion matrix, you will see on what classes the algorithm performs good, and on what classes it performs less good.

7.1. Classify 15-min periods (CNN)

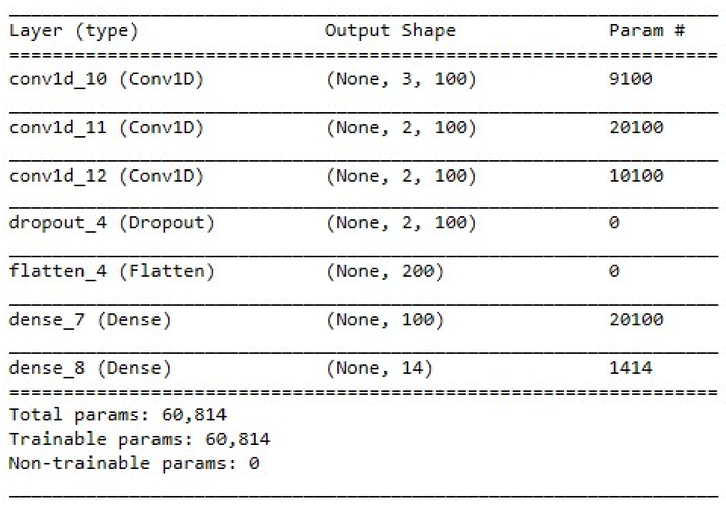
Note: Result only presented for Option 2 Convolutional Neural Network (but code for a fully connected neural network is found in same Jupyter notebook)

Code: Jupyter Notebook "AI_Eband_Model_CNN_4.ipynb"

7.1.1. Classes

0. No event detected
1. Out of service
2. Hardware restarted
3. Stuck on low modulation
4. Unstable Antenna / Wind
5. Intermittent LOS problem
6. LOS or Alignment problem
7. Bad LOS or Alignment problem
8. Affected by Rain
9. Stuck on mid modulation
10. Wet snow
11. Equipment fault
12. Equipment fault far end
13. No or limited data

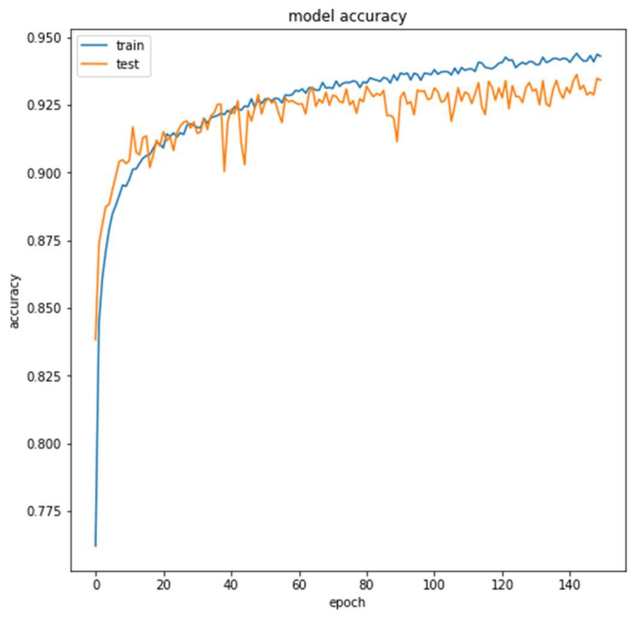
7.1.2. Model

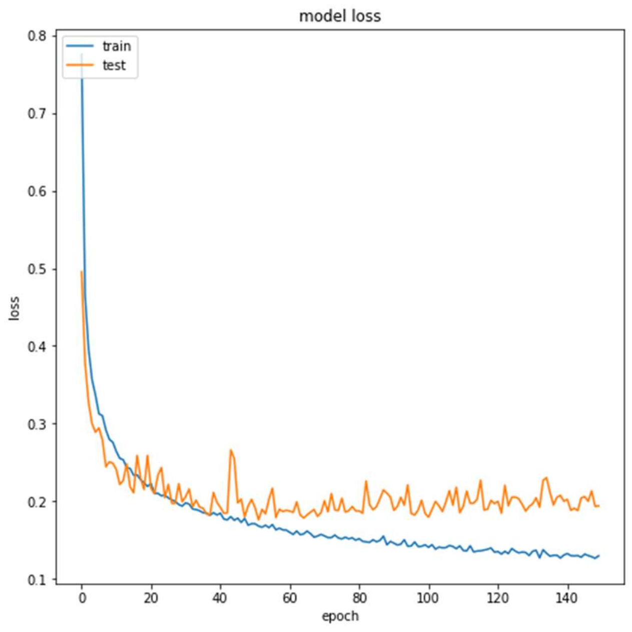


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_10 (Conv1D)           (None, 3, 100)            9100
_________________________________________________________________
conv1d_11 (Conv1D)           (None, 2, 100)            20100
_________________________________________________________________
conv1d_12 (Conv1D)           (None, 2, 100)            10100
_________________________________________________________________
dropout_4 (Dropout)          (None, 2, 100)            0
_________________________________________________________________
flatten_4 (Flatten)          (None, 200)               0
_________________________________________________________________
dense_7 (Dense)              (None, 100)               20100
_________________________________________________________________
dense_8 (Dense)              (None, 14)                1414
=================================================================
Total params: 60,814
Trainable params: 60,814
Non-trainable params: 0
_________________________________________________________________
```

7.1.3. Learning Curve and Accuracy

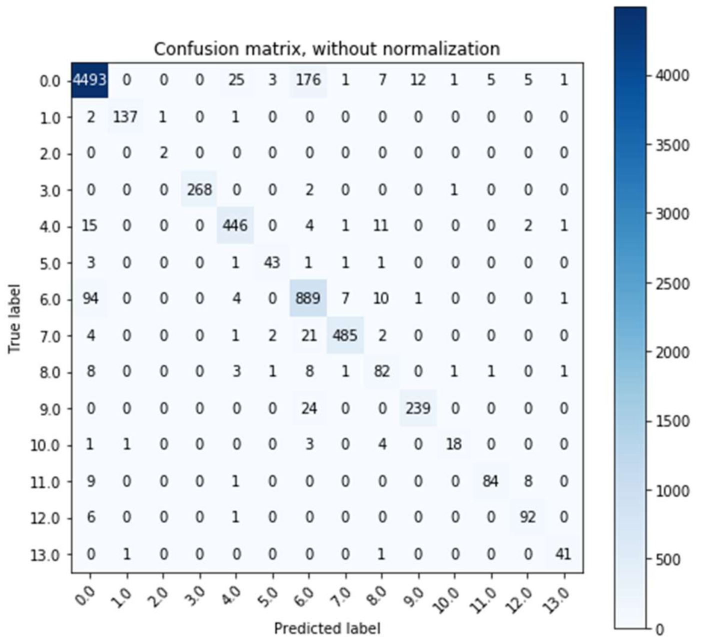
```
Test Accuracy = 0.934
```





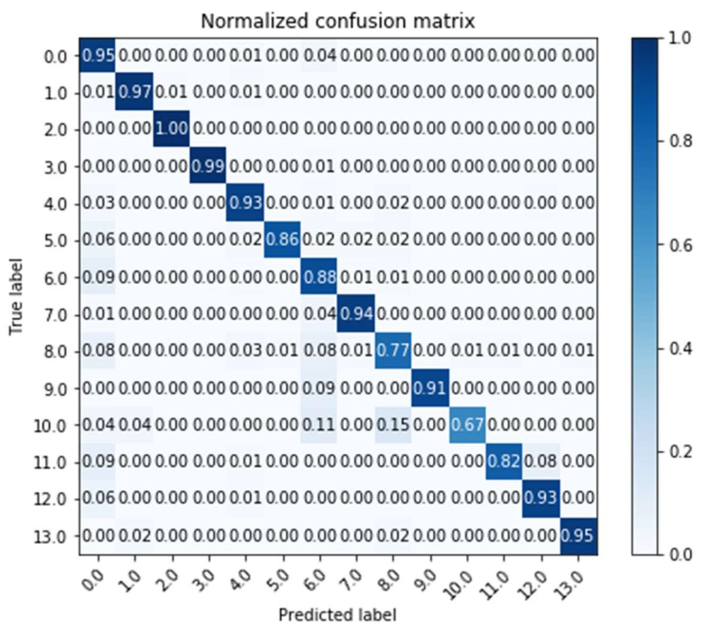
7.1.4. Confusion Matrix

Here you see result and accuracy per class. True label vs. predicted label.



True label \ Predicted label	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0
0.0	4493	0	0	0	25	3	176	1	7	12	1	5	5	1
1.0	2	137	1	0	1	0	0	0	0	0	0	0	0	0
2.0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
3.0	0	0	0	268	0	0	2	0	0	0	1	0	0	0
4.0	15	0	0	0	446	0	4	1	11	0	0	0	2	1
5.0	3	0	0	0	1	43	1	1	1	0	0	0	0	0
6.0	94	0	0	0	4	0	889	7	10	1	0	0	0	1
7.0	4	0	0	0	1	2	21	485	2	0	0	0	0	0
8.0	8	0	0	0	3	1	8	1	82	0	1	1	0	1
9.0	0	0	0	0	0	0	24	0	0	239	0	0	0	0
10.0	1	1	0	0	0	0	3	0	4	0	18	0	0	0
11.0	9	0	0	0	1	0	0	0	0	0	0	84	8	0
12.0	6	0	0	0	1	0	0	0	0	0	0	0	92	0
13.0	0	1	0	0	0	0	0	0	1	0	0	0	0	41

0. No event detected
1. Out of service
2. Hardware restarted
3. Stuck on low modulation
4. Unstable Antenna / Wind
5. Intermittent LOS problem
6. LOS or Alignment problem
7. Bad LOS or Alignment problem
8. Affected by Rain
9. Stuck on mid modulation
10. Wet snow
11. Equipment fault
12. Equipment fault far end
13. No or limited data



True label \ Predicted label	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0
0.0	0.95	0.00	0.00	0.00	0.01	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.0	0.01	0.97	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.0	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3.0	0.00	0.00	0.00	0.99	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4.0	0.03	0.00	0.00	0.00	0.93	0.00	0.01	0.00	0.02	0.00	0.00	0.00	0.00	0.00
5.0	0.06	0.00	0.00	0.00	0.02	0.86	0.02	0.02	0.02	0.00	0.00	0.00	0.00	0.00
6.0	0.09	0.00	0.00	0.00	0.00	0.00	0.88	0.01	0.01	0.00	0.00	0.00	0.00	0.00
7.0	0.01	0.00	0.00	0.00	0.00	0.00	0.04	0.94	0.00	0.00	0.00	0.00	0.00	0.00
8.0	0.08	0.00	0.00	0.00	0.03	0.01	0.08	0.01	0.77	0.00	0.01	0.01	0.00	0.01
9.0	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.91	0.00	0.00	0.00	0.00
10.0	0.04	0.04	0.00	0.00	0.00	0.00	0.11	0.00	0.15	0.00	0.67	0.00	0.00	0.00
11.0	0.09	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.82	0.08	0.00
12.0	0.06	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00
13.0	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.95

7.1.5. Comment

On live data - This model performs less good on classes “10-Wet snow”, “8-Rain” and “6-LOS problem”. It is as expected since those problems/ classes require information from more time-steps. This model should be use together with the “24-hour model”.

The unbalanced class distribution has been disregarded (some classes have few training examples). No class weighting has been done but could be considered in future models.

7.2. Classify 24-hour periods (RNN LSTM)

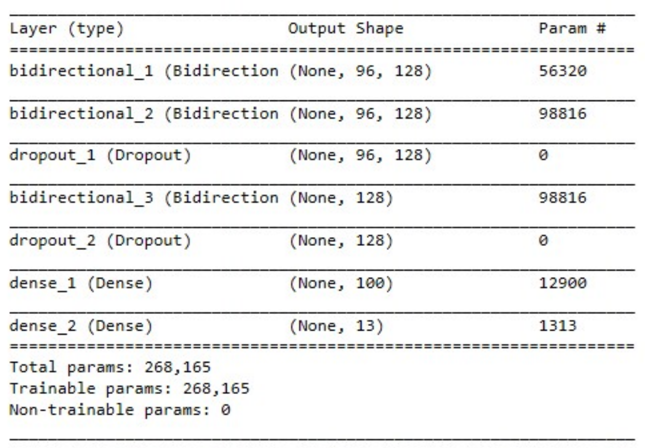
Code: Jupyter Notebook "AI E-band Model_LSTM_96_24.ipynb"

7.2.1. Classes

0. OK
1. The antenna is Unstable
2. Wet Snow stuck on antenna
3. Affected by Rain
4. *Line of sight or Alignment problem*
   (Class Not used for now)
5. Serious Line of sight or Alignment problem
6. *Intermittent Line of sight problem*
   (Class Not used for now)
7. The link is stuck on low modulation
8. Equipment problem (errors)
9. Out of service detected
10. Limited data. Cannot do reliable predictions
11. A hardware restart was detected
12. The link is stuck on mid modulation

Note that numbering of classes differs from the 15-min model.

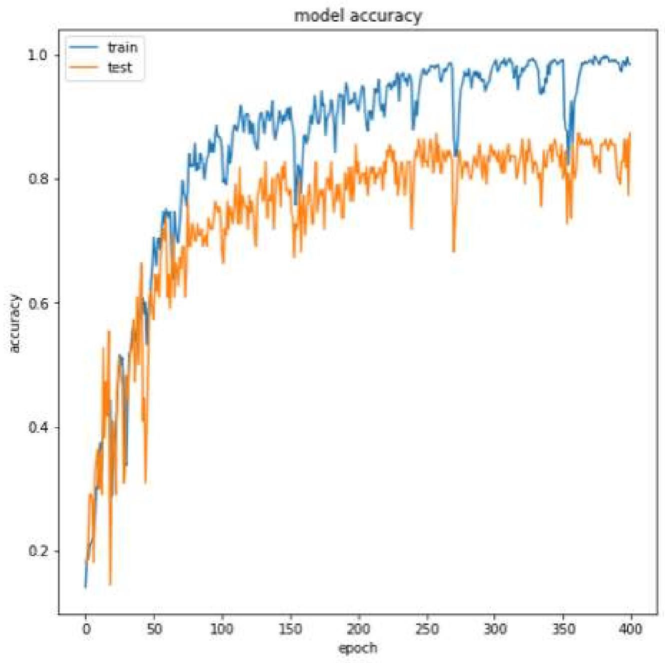
7.2.2. Model

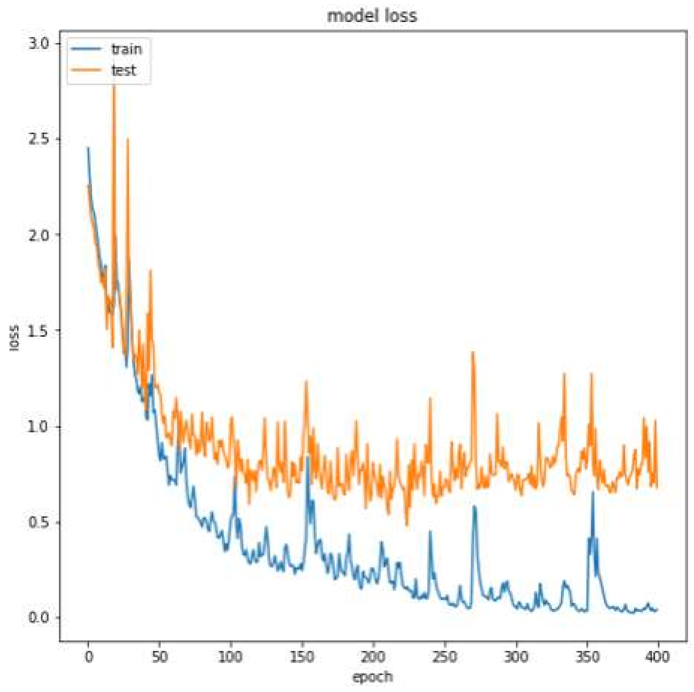


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_1 (Bidirection (None, 96, 128)           56320
_________________________________________________________________
bidirectional_2 (Bidirection (None, 96, 128)           98816
_________________________________________________________________
dropout_1 (Dropout)          (None, 96, 128)           0
_________________________________________________________________
bidirectional_3 (Bidirection (None, 128)               98816
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               12900
_________________________________________________________________
dense_2 (Dense)              (None, 13)                1313
=================================================================
Total params: 268,165
Trainable params: 268,165
Non-trainable params: 0
_________________________________________________________________
```

7.2.3. Learning Curve and Accuracy

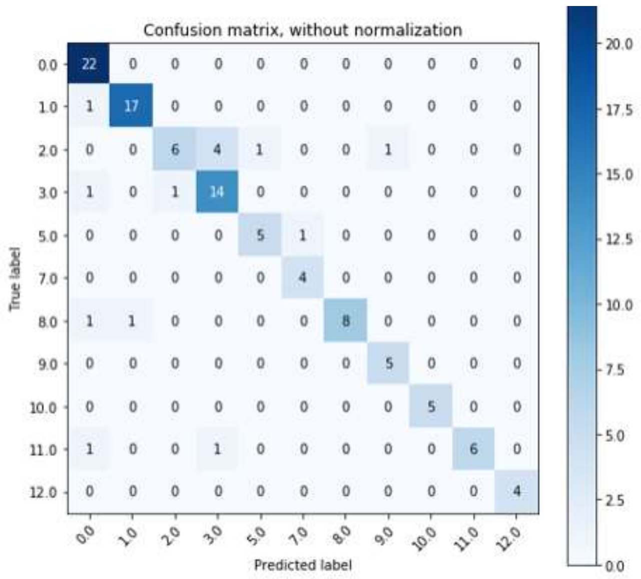
```
Test Accuracy = 0.879
```

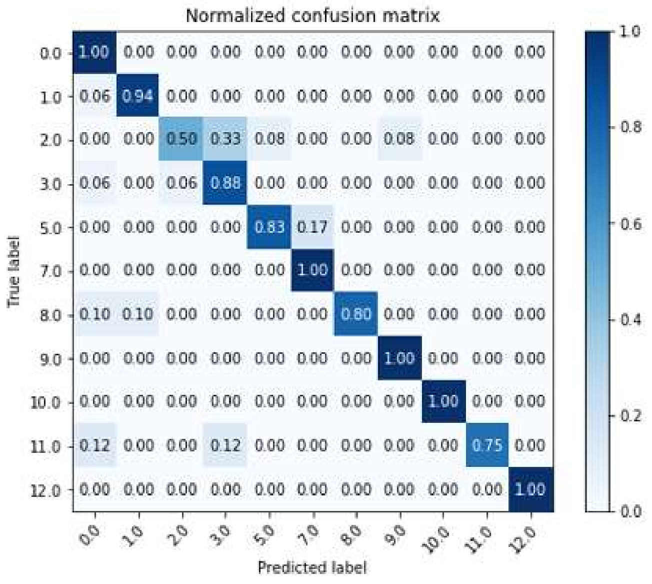




7.2.4. Confusion Matrix

Here you see result and accuracy per class. True label vs. predicted label.





0. OK
1. The antenna is Unstable
2. Wet Snow stuck on antenna
3. Affected by Rain
4. *Line of sight or Alignment problem (Class Not used for now)*
5. Serious Line of sight or Alignment problem
6. *Intermittent Line of sight problem (Class Not used for now)*
7. The link is stuck on low modulation
8. Equipment problem (errors)
9. Out of service detected
10. Limited data. Cannot do reliable predictions
11. A hardware restart was detected
12. The link is stuck on mid modulation

7.2.5. Comment

This model looks promising. It can be overfitted up to 99% accuracy on training set. With more data I expect this model to get high accuracy on test set.

- Regarding class "8-Equipment problem" there are too few training examples, to get a good feeling for how accurate the classification really is here. (Many train examples augmented)

This model could work good for overall root cause analyses and summary reports.

8. FINDINGS

8.1. What can be done to get better results?

1) **More features.** Below features is on my wish list. (Have been requested to vendor..)
   a. Max and Min "Radio received signal level" for each 15 min period
   b. Max and Min "Signal to Noise Ratio" for each 15 min period

   With above features, I believe we will get improved results on predictions. We might also be able to add extra classes, e.g. 'interference problems'.

2) **More data**. Deep learning requires a lot of training data. With more data we can build better and more accurate models.

3) *** Better Neural Network architectures**. I'm sure there are experts that can help here.

   * Some of the events we want to classify on 15 min level require information from many previous times steps. One example is errors due to snow where pattern usually must be evaluated over a long period of time. Best model seems to be a bidirectional lstm recurrent neural network but a convolutional neural network, that looks at more timesteps, could work well too. Currently there are too few training examples to train a lstm RNN that classify each 15 min period.

4) **Build parallel machine learning models**. When we get many classes, it might be better to use different models for different classes as different classes require different features.

5) **Better resolution on performance data**. With better resolution we will get better results. Currently we only have 15 min data available from our management system. A perfect model would have resolution < 1 minute with streaming data to predict on. That would of course generate a lot of data and create other kinds of problems.

6) **Streaming performance data from network elements.** This prototype works on yesterday's data to classify events and to do 'auto root cause analyses. With streaming data, we could have real time classifications on actual and last 15-min, last hour, last 8 hour, last 24 hours, and so on.

7) **Build our own data collector.** It is a good idea to build our own data collector. If we do this, we are not dependent on a system vendor. One big advantage would be to collect neutral data. It means that we can cooperate and share data between operators. - Even if we don't use the same kind of equipment (read same system vendor)

9. LIMITATIONS

Please note that this is a prototype.

1) The Algorithm contains a lot of bias since I have labeled the whole training set myself. I have also augmented parts of the training set.
2) The prototype is today in use at Hi3G Denmark for testing purposes. It performs good on basic events but still makes mistakes where it needs to be guided. With guiding I mean generating new training examples.
3) The algorithm is not production ready yet, (e.g. it cannot generate actionable information automatically). To perform well (and for engineers to trust the machine) the accuracy must be better (read excellent). This I believe could be accomplished by further development as mentioned in "Findings" above. Getting the suggested features would be a great starting point, and I think that vendors understand the importance of providing those.

10. CONCLUSIONS

We don't want engineers to spend time on comparing graphs and excel sheets to understand how the network perform. Its better to train a machine to do this. And it will be a lot cheaper.

This article started with a question:

Can we train an algorithm to do event classification on E-band links – by using existing performance data?

Yes – we can do basic classification to get an overview of what cause errors and outage on E-band links. We can get top lists per class – e.g. links with unstable antennas.

But – currently, we cannot classify with so high accuracy, that we can replace the manual root cause analyses. To do that we need to improve the model and algorithm. My ideas are listed under "Findings".

Finally: Labeling takes time and is hard work. Domain expertise is crucial to build a successful product. You need experienced engineers to teach the machine to classify MW links. Just like you need experienced doctors to teach a machine to understand x-ray pictures.

Machine Guidance is the key.

Thanks for reading, and I hope to hear from you soon. Let's give the repetitive and boring work to the Machines.

Glenn Landgren

Malmö, Sweden 2019-12-15