Learning to detect NLOS satellites

Hendrik Serruys

September 2019

1 Introduction

In an urban canyon environment, Non-Line-Of-Sight (NLOS) satellites can be visible to a receiver through reflections of the signal. However, as the signal is reflected, the time required for the signal to reach a receiver is longer compared to a satellite in the same position but within Line-Of-Sight (LOS). The reflection therefore causes the receiver to obtain an increased error on its position estimation. If the receiver could evaluate the incoming signal and (1) classify it as LOS/NLOS or (2) obtain a level of confidence that the signal is LOS, then the receiver should be able to optimise its PVT accuracy, either by selecting a good subset of satellites (hard satellite selection), or by weighing the contribution of a satellite in the position estimation (soft satellite selection).

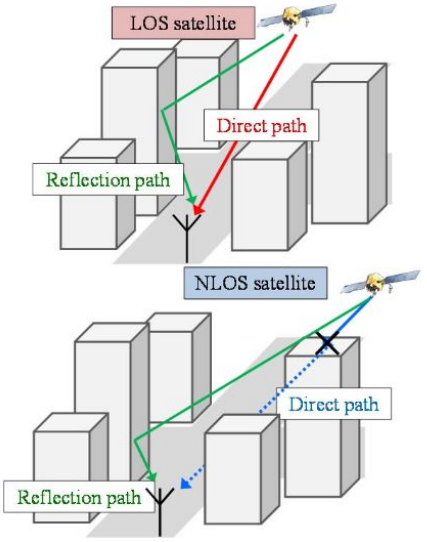


Fig. 1. The multipath and NLOS effects in an urban canyon. (a) Multipath effect, (b) NLOS propagation [1].

[Source: Hsu L., *"GNSS Multipath Detection Using a Machine Learning Approach"*]

As the figure above shows, reflected signals can occur both while the satellite is LOS or NLOS. In this project, we focus solely on separating LOS from NLOS as the labelling of our data is done by a fisheye camera (see project Floor Melman). Therefore, in this setting, reflected signals from LOS satellites are considered noise. It will be part of this project to evaluate the ability of learners to adapt to this LOS/NLOS setting, and whether or not it is vital to obtain better data labelling than those obtained through visual segmentation of the sky.

2 Approach

The first difficulty in our NLOS problem concerns the **input variables**. This will heavily depend on the receiver and the desired application. We should therefore clearly separate the different cases. For instance, we can create models for receivers which calculate pseudoranges based on one frequency band, dual frequency bands, etc. If we want to target an application which is able to use the correlation functions, we will require a software GNSS receiver to acquire these functions in digital form. Other input information like elevation angle, the difference between code and phase measurements, KF innovations... should also be considered. Furthermore, we can consider models specifically designed to process only signals from one GNSS constellation. We will also have to experiment with different normalisation techniques and investigate the impact of other data transformations.

The second difficulty involves the **model** itself. We should evaluate a wide range of models (i.e.: SVMs, basic feedforward NNs, KNN, Decision Trees (bagging, boosting...), ensembles...).

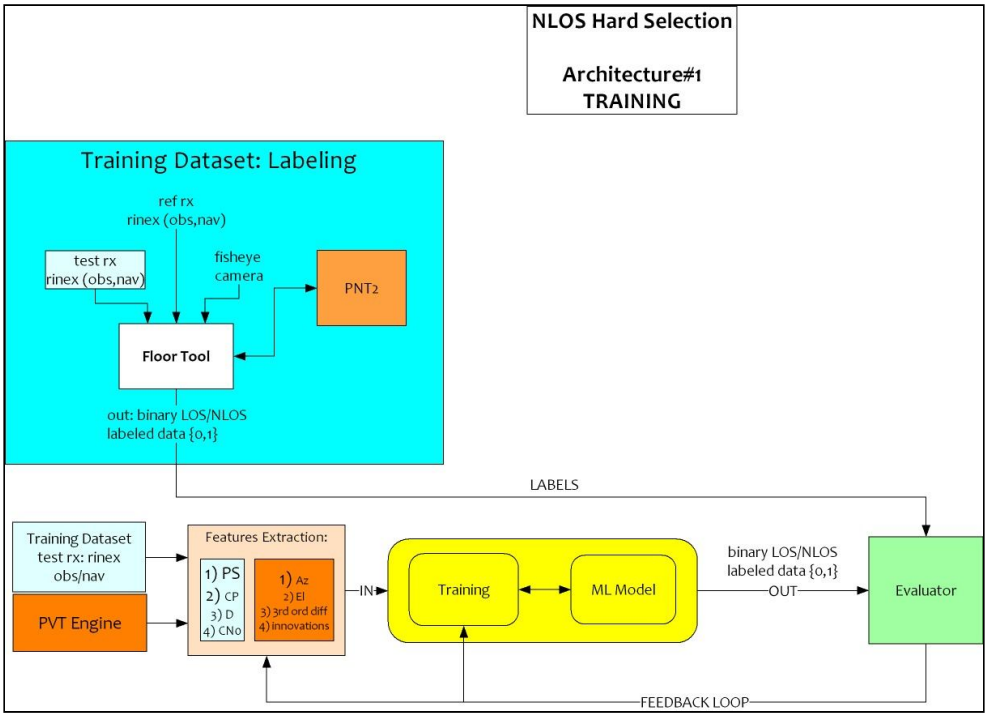
The third difficulty involves the desired **output** and corresponding error measurement which is used to optimise the model. I believe the two straightforward output settings are:

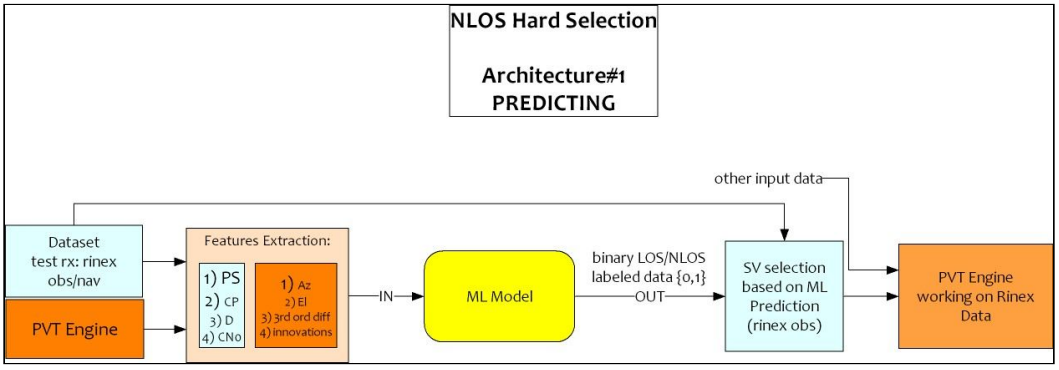
- Hard classification, where we simply desire a binary True or False as to whether the input signal is LOS.
- Soft classification: where we apply a softmax to the network output to obtain the probability or belief of the network in that the input signal is LOS. This setting might be interesting as we could provide weights to compute the position respective to the network belief about the received signals. Where the hard classifier classifies two signals as LOS, the soft classifier might show that for one of these signals the network is in fact very uncertain about that decision (e.g. 55% chance of LOS).

The fourth difficulty involves **measuring the performance** of the network. Some common metrics like accuracy, precision, recall, F1 can be used. But we could also evaluate classification performance by measuring positioning accuracy if we are able to tell the position calculation module to use the network output in selecting the satellites to estimate PVT.

Lastly, the fifth difficulty is concerned with **data selection**. If we have data from multiple tours, we can recognise them as different data sets and evaluate all our models separately for every dataset. We could also merge all datasets and not care about different tours. Furthermore, we could train the model on one tour and test it on another. It will be important to evaluate whether the available data is sufficient towards training a general learning algorithm.

The specific architectural components dedicated to addressing these problems are visualised in the following illustration.





We focus on the supervised learning setting. We experiment with different input variables which originate either from observational data or navigational data. The latter requires a PVT engine to obtain. By use of a fisheye camera, labels are obtained for each Space Vehicle (SV). The labels inform us if the SV is either LOS (SV appears in sky segment of image) or NLOS (SV appears to be behind an obstacle). We train our learners for hard classification. If the learner is able to make accurate predictions on the label of an SV, we can deploy the learner within a PVT engine and use it for hard satellite selection before PVT estimations. The benefit of this approach is it relative simplicity. The learner can be trained fully outside of the PVT engine as it is not aware of any PVT calculations. Once trained, the learner can be used to aid any receiver as long as the required inputs of the learner can be produced. The main weakness of this approach is that the learner is not aware of the context (group of satellites that are processed simultaneously within a PVT engine) and is not aware of the ultimate goal (optimising PVT estimation), and might therefore classify all satellites as NLOS, effectively causing the satellite selection process to eliminate all satellites in view.

3 Data

3.1 Datasets

Currently, four datasets are available corresponding to four tours. The datasets do not fully correspond to the raw RINEX measurement files. This is because the camera labelling was only done for subsets of the data. I believe the main reason was the long processing duration, as well as the fact that the labelling project is still going on and needs to be further improved.

Dataset Name	**Duration of trip (HH:MM:SS)**	**Epoch rate (epochs/s) [1]**	**Number of labelled satellites [2]**	**Fraction LOS [3]**	**Fraction NLOS [4]**
AMS_01	03:00:00	1	203605	0.65	0.35
AMS_02	01:09:45	1	80274	0.71	0.29
ROT_01	02:56:59	1	173368	0.64	0.36
ROT_02	01:59:59	1	123381	0.69	0.31

[1]: Number of epochs per second.
[2]: Per epoch, a variable number of satellites is observed and labelled. This number is obtained by summing over all satellites which received a label over all epochs.
[3]: Fraction LOS = $\frac{\#LOS\ labels}{\#LOS\ labels + \#NLOS\ labels}$
[4]: Fraction NLOS = $\frac{\#NLOS\ labels}{\#LOS\ labels + \#NLOS\ labels}$

3.2 Cleaning the data

There are a lot of NaN values within the data. It is important to clearly understand their cause - which will be different for different variables - and how to handle them. Most commonly, we will either remove these data records entirely, or replace the NaN with a sensible value.

Description	**Current treatment**	**Additional info**
We obtain data for all satellites, even those which are not in view.	Remove record for not observed satellites (e.g. below horizon)	

Some satellites have observable data and therefore are in view, but lack a LOS/NLOS label from the camera. This is mainly caused due to the camera itself, which might not be able to see a satellite (close to the horizon), although the receiver does obtain a received signal.	Remove record	
Due to loss of lock, there are records without carrier phase information.	As the lack of information is important (indicator for loss of lock), we replace the NaN value with 0.	Replacing NaN with 0 is problematic later when we want to normalise the data. (0s will have a big influence on average and standard deviation). We need a different treatment OR only standardise non-zero values. Currently, min-max normalisation is applied over all non-zero values. Furthermore, an extra binary input variable could be defined, indicating a carrier phase measurement is observed (“1”) or not (“0”) .
There is no innovation information for the first timestep (as for the first timestep, LS is used and not the KF) and no third order difference information for the first three timesteps.	Remove records of first three seconds	
From the timestep a new satellite becomes observable, again three timesteps are required until third order difference data is available for this satellite.	Remove record with NaN values for third order difference	First, removing these records seem to drastically increase the fraction of LOS labelled satellites (from 61% to 66% for dataset ROT_01). Second, the third order difference (TOD) NaN values represent about 10% of the data (ROT_01). Third, TOD NaN values are a good indication that a satellite hasn’t been observed (continuously) for a long time. These three facts indicate we should find a better solution to treat NaNs here.

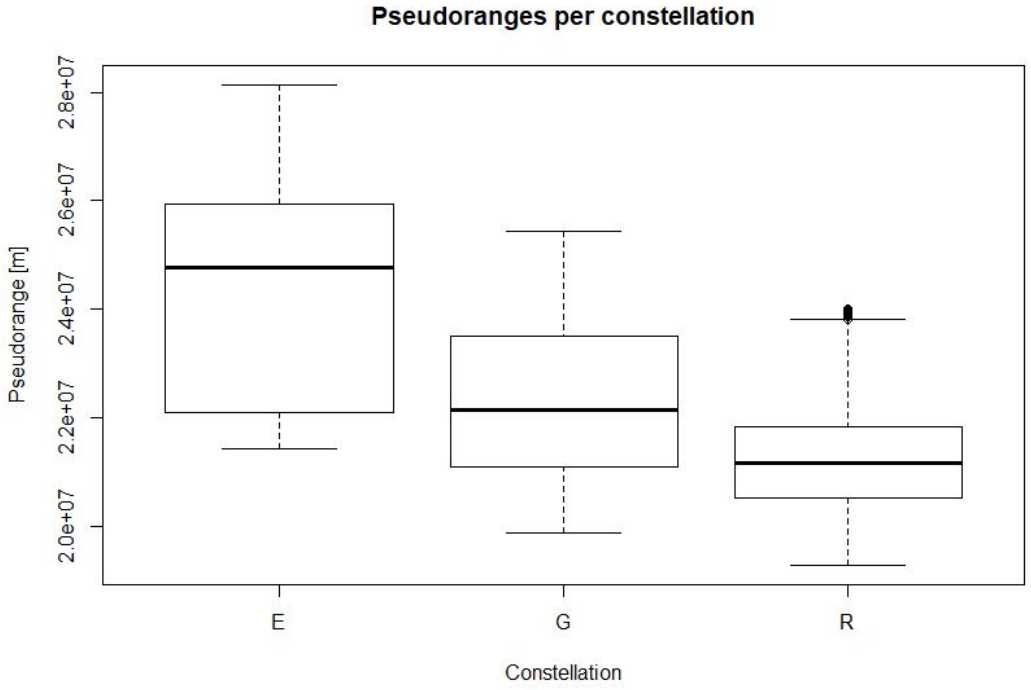
		Best to give them a large value (+inf) or a value of 0?
At times there is no innovation available for an SV. Floor believes the cause could be that there is a small elevation threshold to select satellites for the effective PVT calculations. Therefore, low elevation SVs are discarded and have no innovation.	Remove record	Innovation NaN values represent about 1% of the ROT_01 dataset.

3.3 Exploratory statistical analysis

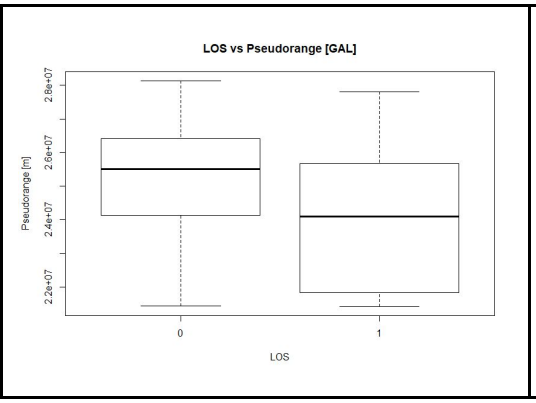
The following analysis is based on the **ROT_01** dataset. An exploratory statistical analysis is a vital step in machine learning and data mining in general. Insight into the raw input data variables can uncover important biases in the dataset or outlying data records which may need to be removed. Furthermore, strong correlations between variables are important to discover and suggest that the overall complexity of the input data can be reduced. Finally, we should have a general idea of how the predictor variables are related to the response variable

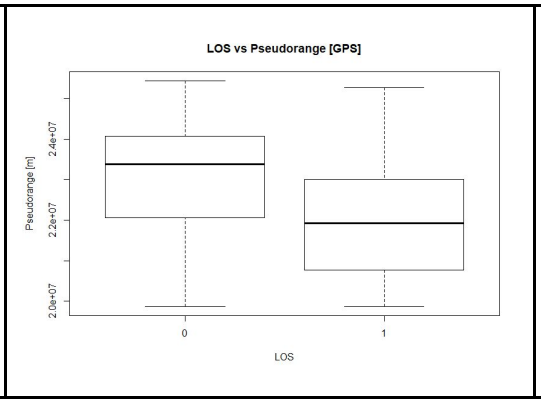
Observable variables

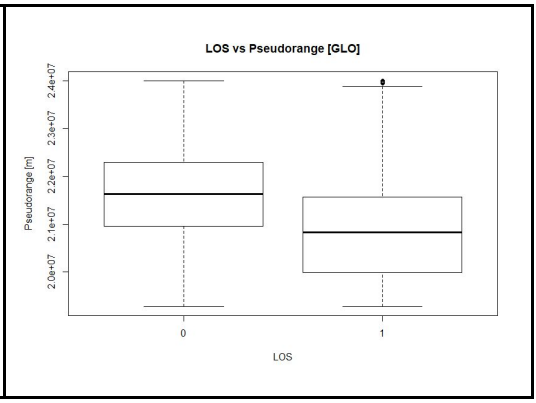
Pseudorange



As expected, we find the pseudoranges vary per constellation. What is interesting however is that there is more variance for GAL and GPS (larger IQR).







As there is a clear relation between pseudoranges and LOS classification, the first plot suggests we should consider a different learner per constellation, normalise the data per constellation or at least provide a one-hot encoding of the constellation as input. This should increase the learner's ability to learn patterns between the pseudorange input variable and the labelling. If we do not somehow communicate the constellation to the learner, the learner will have an incentive to label GLO signals as LOS and GAL signals as NLOS.

Carrier phase

In general, similar observations can be made for carrier phase measurements as were made for the pseudorange.

However, carrier phase measurements are often not present due to loss of lock (LL). The fraction of LL compared to the number of observations per constellation is shown in the table below. We observe that LL is not an uncommon phenomenon.

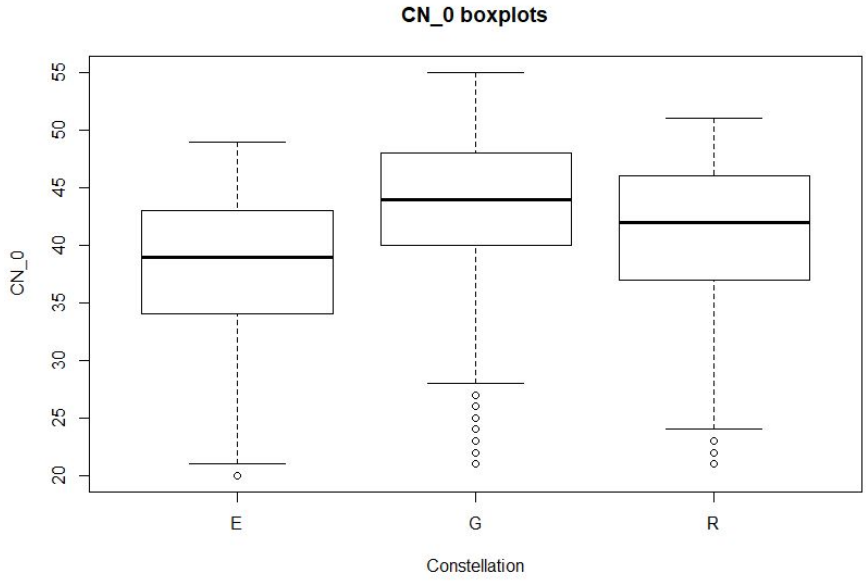
	GAL	**GPS**	**GLO**	**All constel.**
fraction LL	0.18	0.16	0.22	0.19

In the next table, we investigate the relation between LL and LOS:

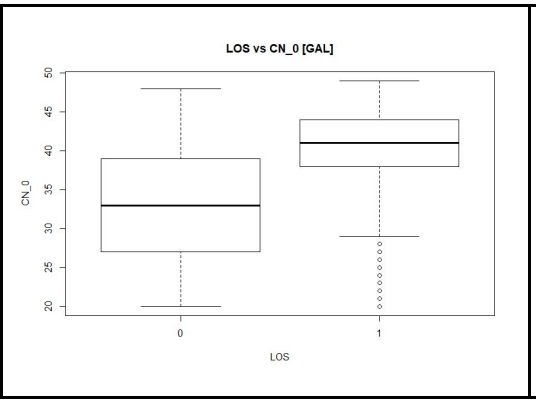
	GAL	**GPS**	**GLO**	**All constel.**
P[LOS | no LL]	0.7651	0.7583	0.7789	0.7635
P[LOS | LL]	0.1780	0.1855	0.2245	0.1999
$\frac{P[LOS \mid no\ LL]}{P[LOS \mid LL]}$	4.2983	4.0879	3.4695	3.8194

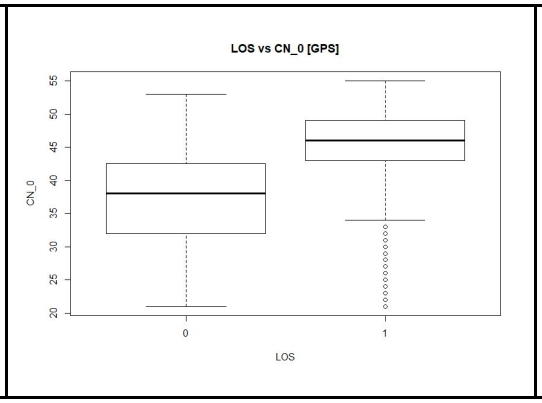
There appears to be a strong correlation between LOS and LL. The probability for an SV to be LOS when a carrier phase is observed is almost four times as large as when there's a loss of lock. We should therefore be interested in defining a new binary input variable, indicating whether or not carrier phase information is available.

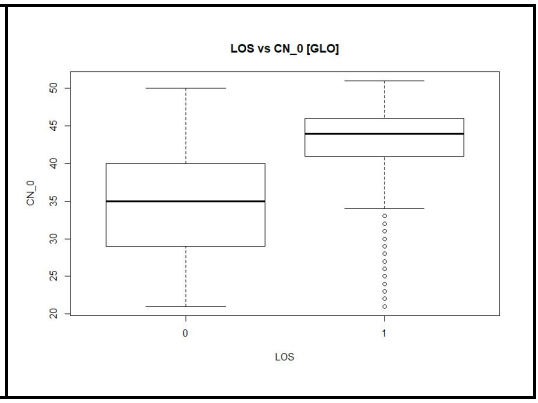
Carrier to Noise Ratio (CN0)



We observe clear differences in CN0 for different constellations.

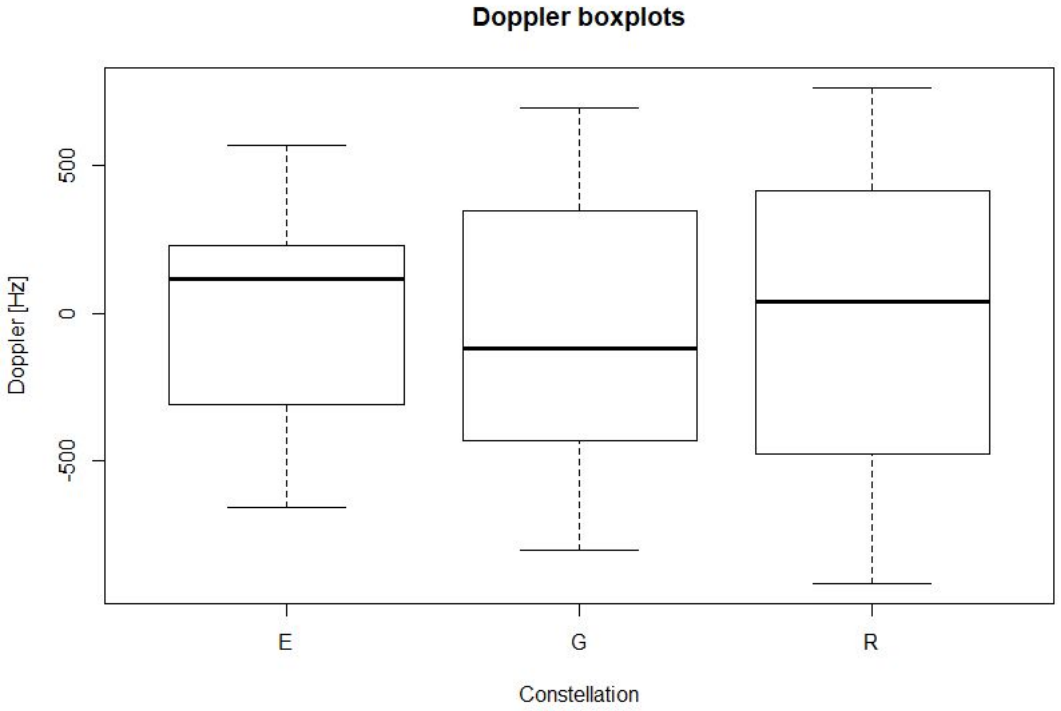






As expected, the average CN0 for LOS SVs is significantly higher than for NLOS SVs. Therefore, once more we find the constellation to be an important input parameter for the learner, as average CN0 per constellation differs.

Doppler

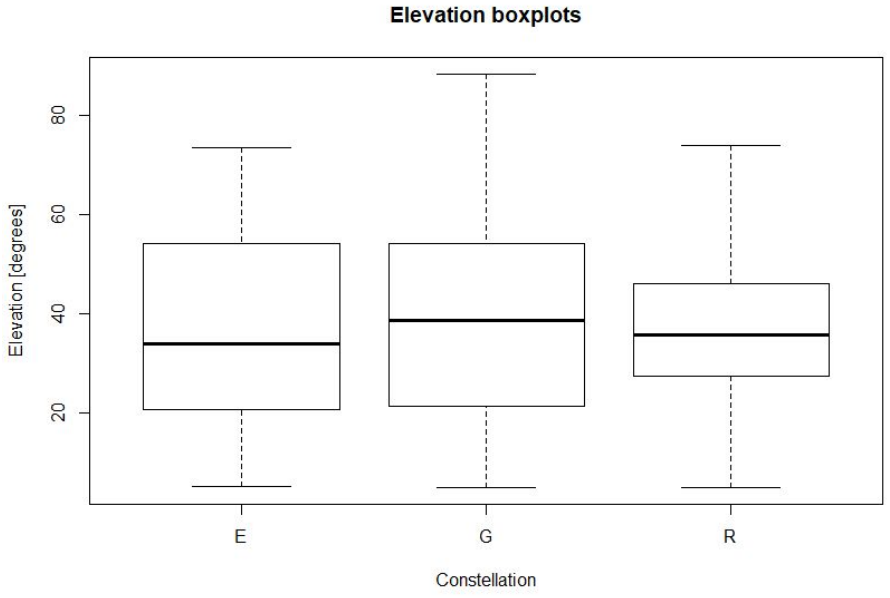


PVT engine variables

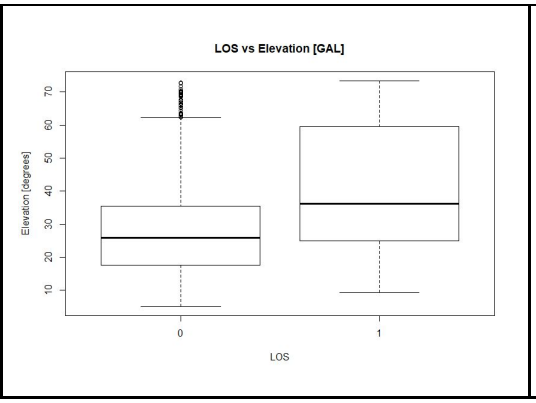
Azimuth

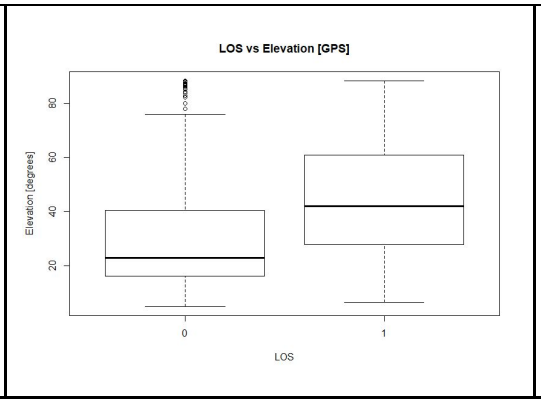
As the azimuth can not be generalised towards predicting an SV to be LOS/NLOS, it should not be provided as an input to the learner.

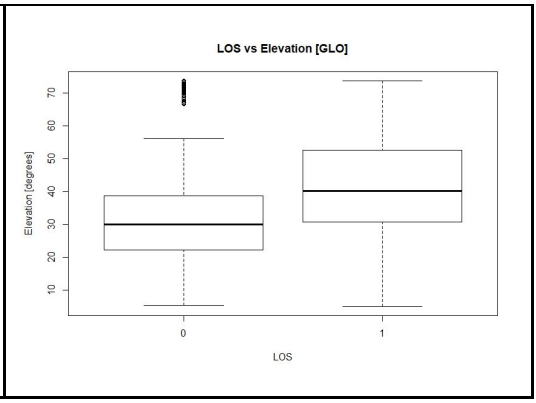
Elevation



Although there are small differences between the elevations per constellation, we should be aware that these differences are caused by random effects (due to the time and place of the obtained data), and therefore on average we want these differences to be minimal. This is important to not lead a learner on a wrong path by providing patterns which are merely circumstantial and will not aid the learner in a new situation.

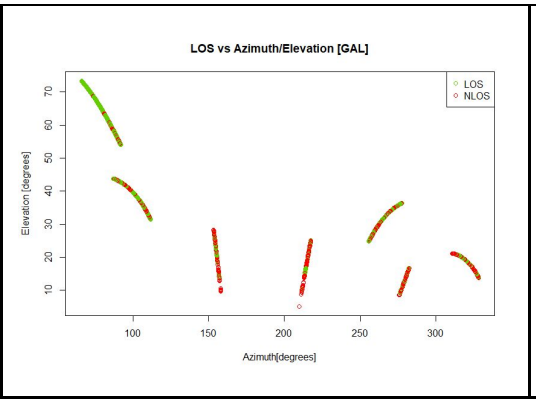


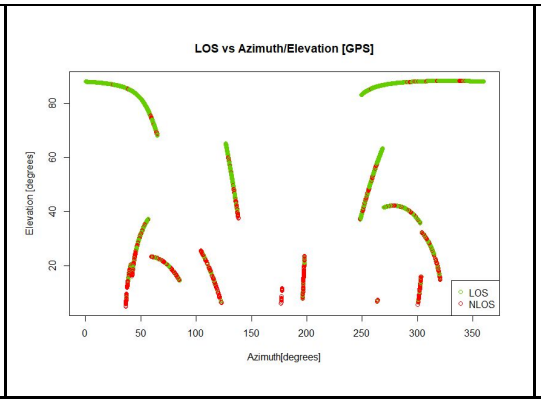


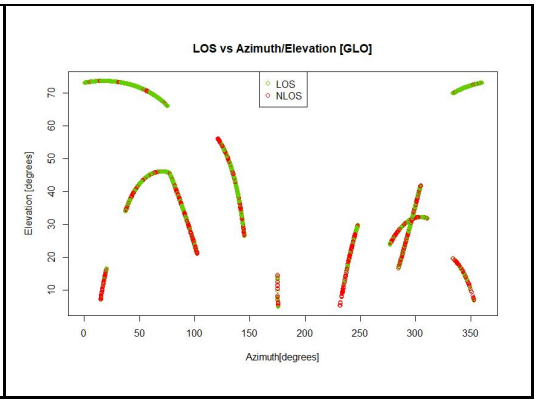


With respect to the LOS/NLOS classes, we can verify the theoretical expectation that LOS SVs tend to have a higher elevation. Furthermore, these plots may be useful to investigate whether the outlying SVs classified as NLOS (LOS = 0) are in fact correctly classified.

Another way to visualise the elevation (and azimuth) to LOS/NLOS:

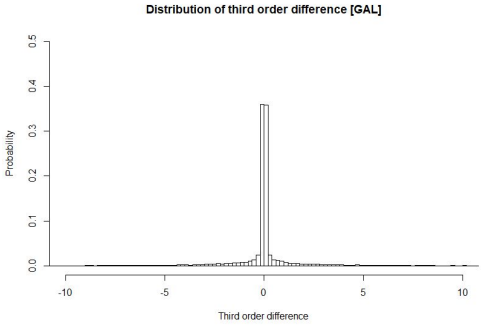


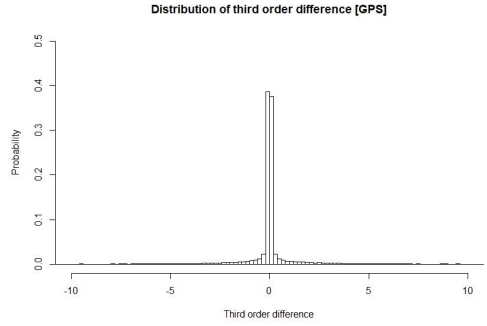


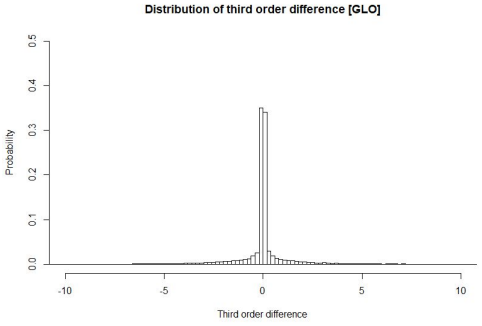


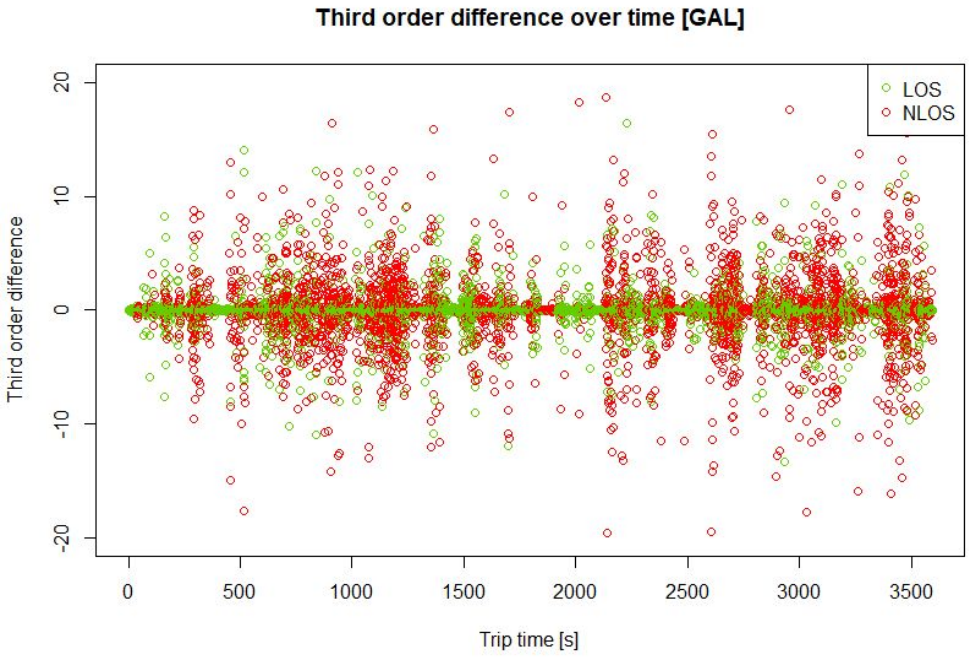
Third order difference (TOD)

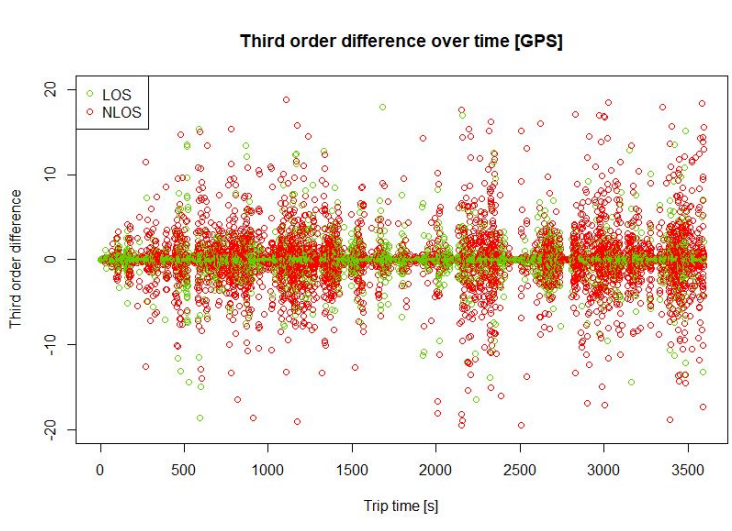
The TODs are strongly distributed around zero, and we find no strong evidence here that one constellation is performing worse than another.

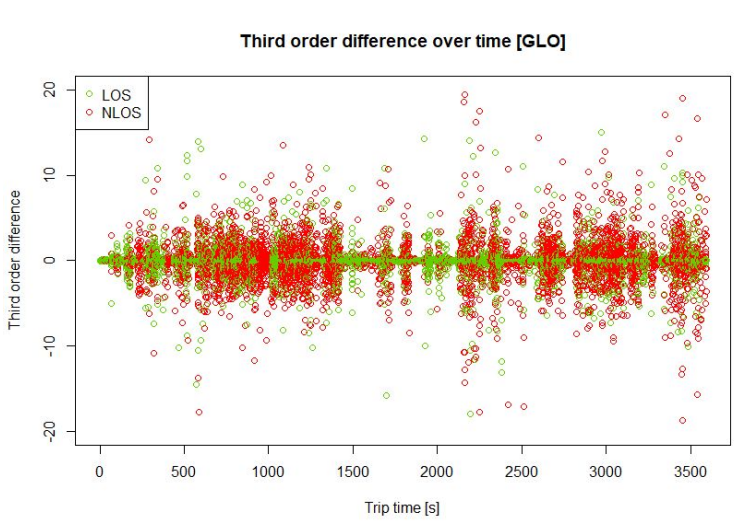

Distribution of third order difference [GAL]
Probability
Third order difference


Distribution of third order difference [GPS]
Probability
Third order difference


Distribution of third order difference [GLO]
Probability
Third order difference

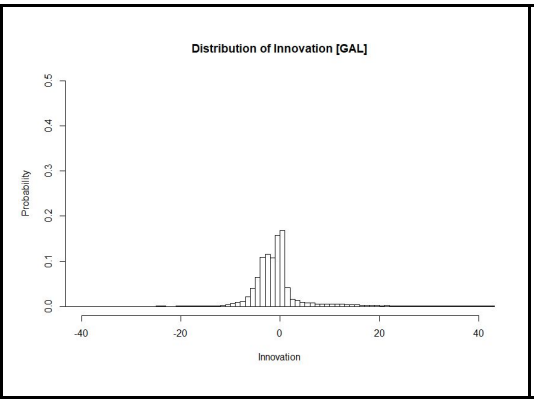

Third order difference over time [GAL]
LOS
NLOS
Third order difference
Trip time [s]

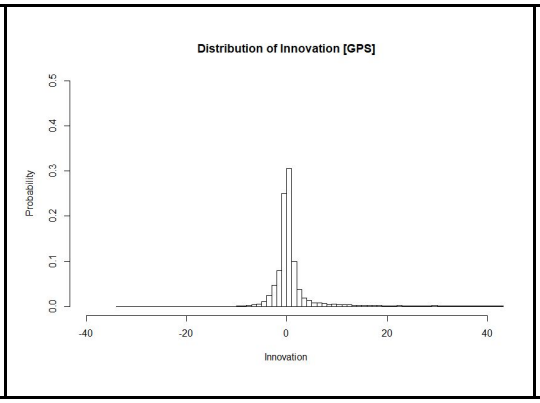

Third order difference over time [GPS]
LOS
NLOS
Third order difference
Trip time [s]

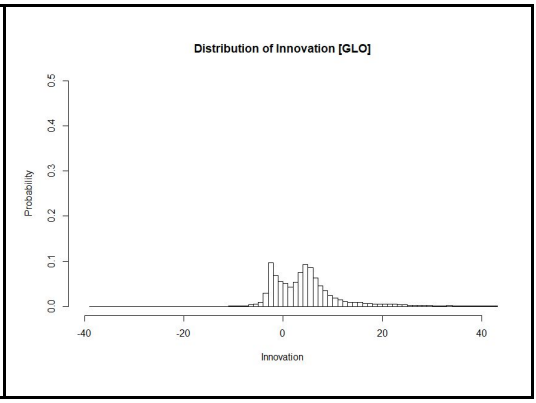

Third order difference over time [GLO]
LOS
NLOS
Third order difference
Trip time [s]

Plotting the TOD over time and separating LOS from NLOS with color, we can observe that LOS SVs tend to have small TODs, while the TOD of NLOS SVs is more strongly dispersed.

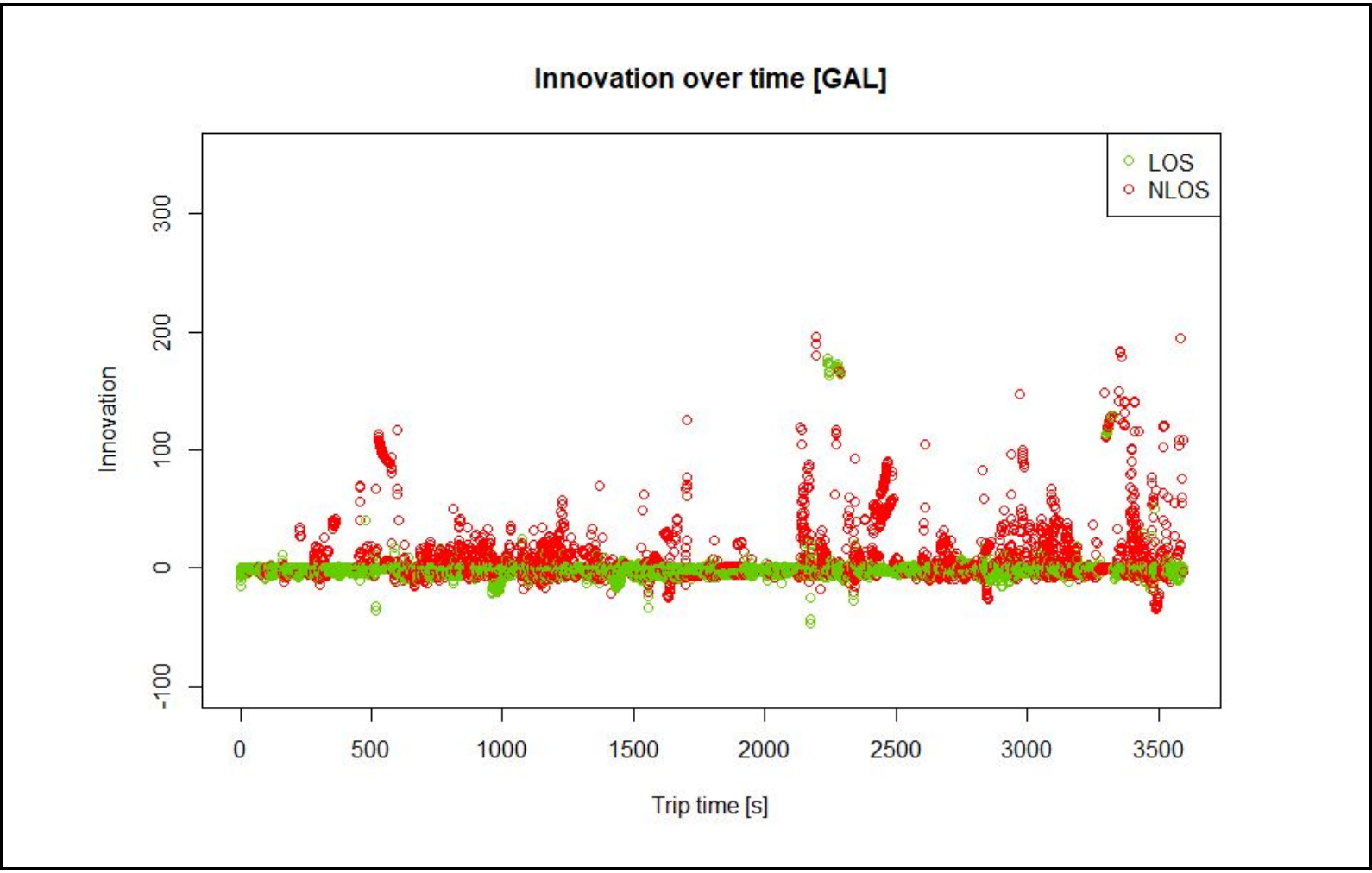
KF Innovations

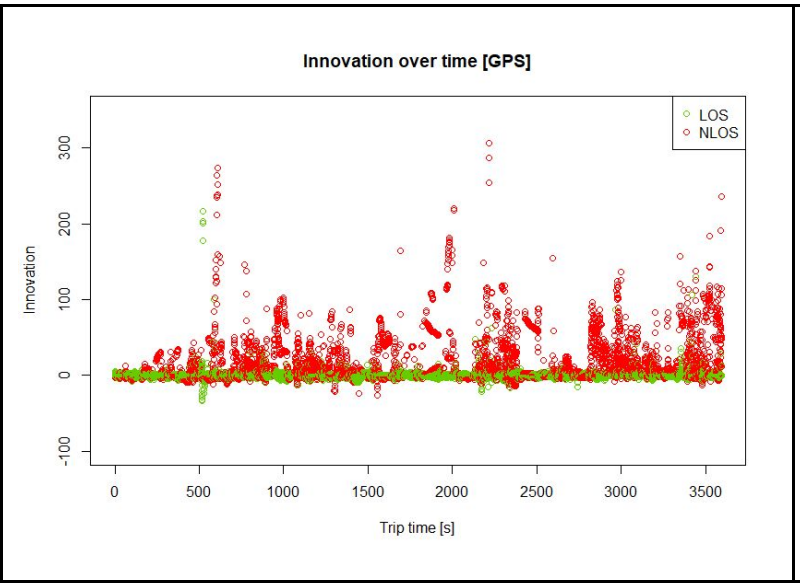


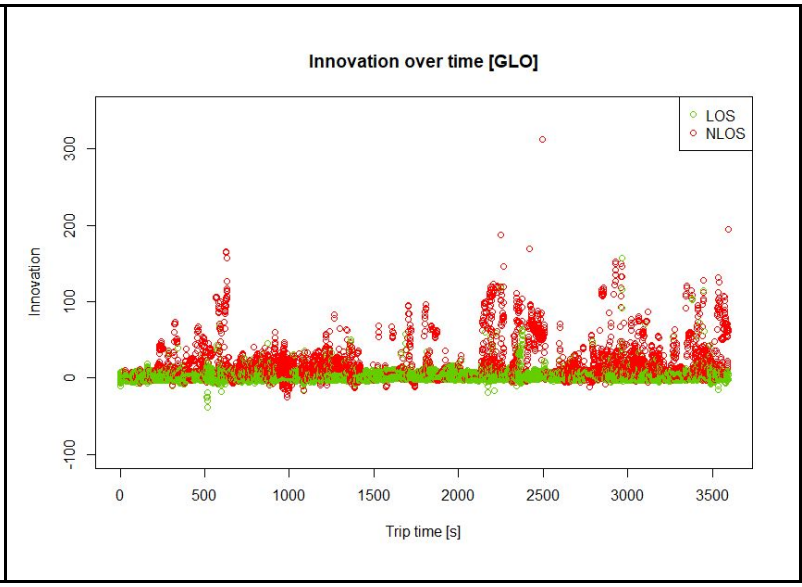




For this tour (ROT_01), strong differences are observed w.r.t. KF innovations per constellation. If these differences are largely caused due to circumstances unrelated to the constellation, then this might lead the learner to not generalise well and perform worse on different tours.



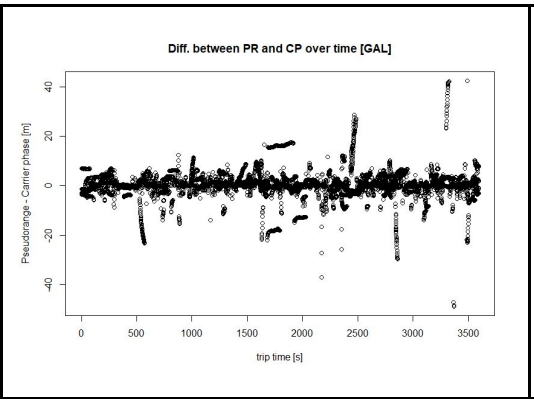


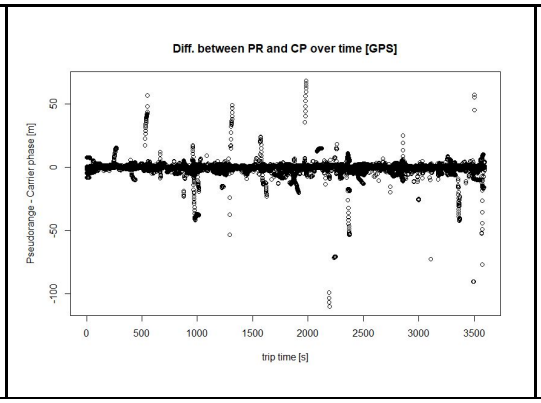


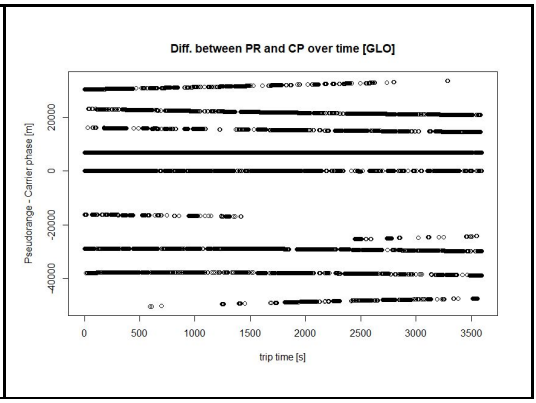
Observing the LOS/NLOS labelling, we find that LOS SVs tend to have a small KF innovations. Furthermore, these plots reveal some interesting cases of outlying labels. For the GAL constellation, a group of LOS labels is found with large innovations around 2200 seconds into the trip. A similar group of outlying LOS labels is found for the GPS system

around 500 seconds. We should consider to remove these data points from the dataset, as they might reflect errors from the camera labelling tool.

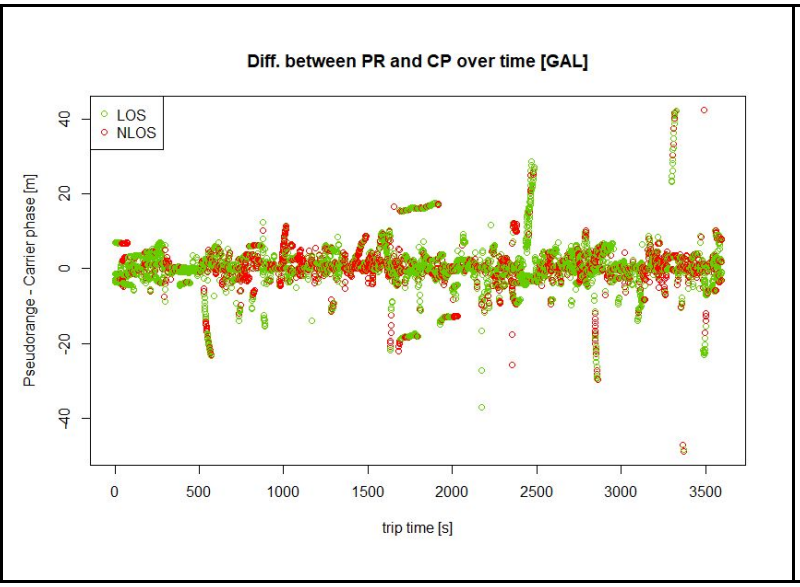
Pseudorange to carrier phase difference

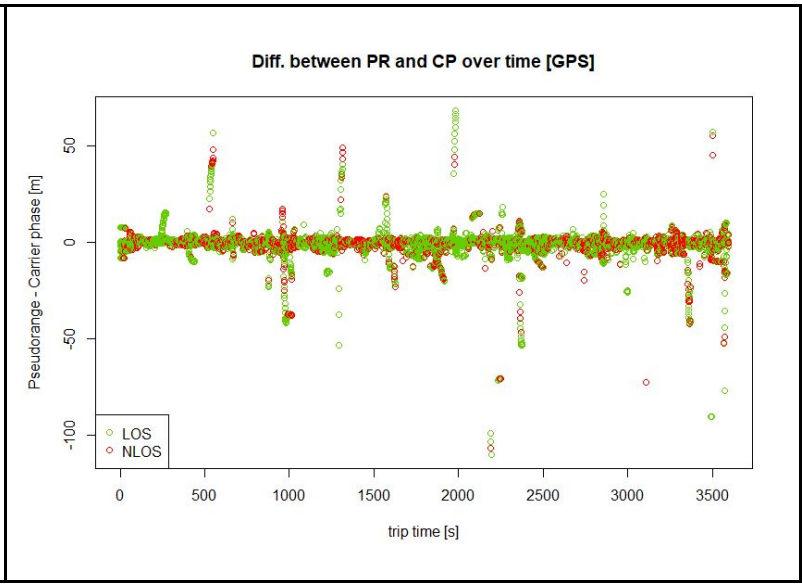






These plots are generated only for data points who have both pseudorange and carrier phase observations. Looking at the difference between pseudorange and carrier phase measurement as the trip progresses, we see that the GLO system has a strongly different behaviour than the other two. This is because GLO uses FDMA instead of CDMA like GPS and GAL. We have to be aware of such differences within the data if we want to build features which can be generalised over all constellations.





Adding the LOS/NLOS labels, we observe no true correlation between the difference of pseudorange and carrier phase with the target classes. It could be concluded that this difference does not provide a good feature. However, this could be investigated further when evaluating the learners.

4 Evaluation

4.1 Evaluation Metrics

Measuring how well a learner is performing is essential, yet non-trivial. Having a good understanding of the different performance areas and how to evaluate them will help us adapt and improve our learners in the appropriate way. As a basic example: if the cost of misclassifying an NLOS signal as LOS is significantly higher than a LOS signal as NLOS, then we'll need to develop ways to bias our learners to avoid false LOS decisions, even if it means increasing false NLOS decisions.

In this section, I will briefly discuss the most common evaluation metrics for classification and their meaning.

True-LOS, False-LOS, True-NLOS, False-NLOS

The **True-LOS** of a classification is the count of LOS signals that were truthfully classified as such by the learner. The **False-LOS** is the count of NLOS signals which were wrongfully classified as LOS.

$$\textbf{True-LOS} = \#\,Signals\ correctly\ classified\ as\ LOS$$
$$\textbf{True-NLOS} = \#\,Signals\ correctly\ classified\ as\ NLOS$$
$$\textbf{False-LOS} = \#\,Signals\ falsely\ classified\ as\ LOS$$
$$\textbf{False-NLOS} = \#\,Signals\ falsely\ classified\ as\ NLOS$$

Precision

The **LOS-Precision** is the fraction of True-LOS to all signals *classified* as LOS (True-LOS + False-LOS). In other words, the precision gives us the probability that a signal is *in fact* LOS, given that the learner *classified* it as LOS. Precision takes a value between 0 and 1, with 1 being optimal.

LOS-Precision is also known as the **LOS Predictive Value**.

$$\textbf{LOS-Precision} = True\text{-}LOS\ /\ (True\text{-}LOS + False\text{-}LOS)$$

$$\textbf{NLOS-Precision} = True\text{-}NLOS\ /\ (True\text{-}NLOS + False\text{-}NLOS)$$

Recall

The **LOS-Recall** is the fraction of True-LOS to all signals which are *in fact* LOS (True-LOS + False-NLOS). In other words, the recall gives us the probability that a signal is *classified* as LOS, given that it is *in fact* LOS. Recall takes a value between 0 and 1, with 1 being optimal.

LOS-Recall is also known as the **True LOS Rate**.

LOS-Recall = *True-LOS / (True-LOS + False-NLOS)*
NLOS-Recall = *True-NLOS / (True-NLOS + False-LOS)*

Fall-out

The **LOS-Fall-out** is the fraction of False-LOS to all signals which are *in fact* NLOS (False-LOS + True-NLOS). In other words, the fall-out gives us the probability that a signal is *classified* as LOS, given that it is *in fact* NLOS. Fall-out takes a value between 0 and 1, with 0 being optimal.

LOS-Fall-out is also known as **False LOS rate**.

LOS-Fall-out = *False-LOS / (False-LOS + True-NLOS)*
NLOS-Fall-out = *False-NLOS / (False-NLOS + True-LOS)*

To help understanding the difference between Precision, Recall and Fall-out, consider the following example. Let's say we have a dataset with 50 LOS and 50 NLOS signals. If a learner learns nothing and simply classifies every signal as LOS, then

LOS-Precision = 0.5 (of all those identified as LOS, only half are in fact LOS)
LOS-Recall = 1.0 (learner retrieved all LOS signals)
LOS-Fall-out = 1.0 (learner will always misclassify NLOS as LOS)

NLOS-Precision = Undefined (no signal was classified as NLOS)
NLOS-Recall = 0 (learner did not retrieve a single NLOS signal)
NLOS-Fall-out = 0 (learner will never misclassify LOS as NLOS).

F1

LOS-F1 is the harmonic average of LOS-Precision and LOS-Recall**.** Intuitively, its importance is that it allows us to conceive the performance with respect to Precision and Recall in a single metric. F1 takes a value between 0 and 1, with 1 being optimal.

LOS-F1 *= 2 * (LOS-Precision * LOS-Recall) / (LOS-Precision + LOS-Recall)*

Accuracy

The **Accuracy** of a learner is the fraction of all correctly classified signals to all signals. The benefit of this metric is that it provides a general idea of the performance of the learner in a single number. A disadvantage is that it may be too general, potentially hiding skewed performance over different classes.

Accuracy *= (True-LOS + True-NLOS) / (True-LOS + True-NLOS + False-LOS + False-NLOS)*

Confusion Table

A **confusion table** is a good way to visualise different performance measurements of a learning algorithm. In the results discussed in this report, it takes the following form:

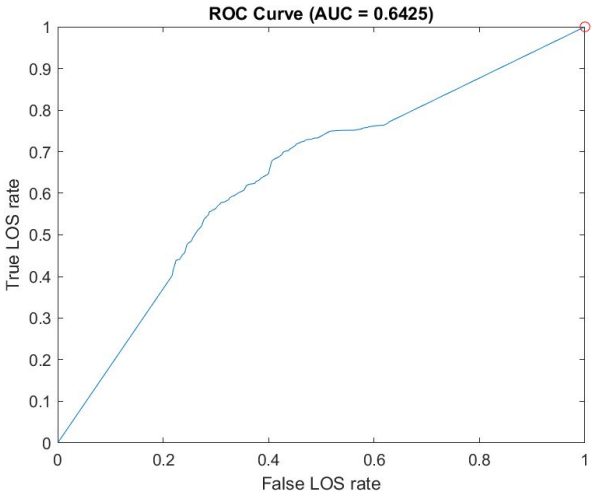
	Factual NLOS	**Factual LOS**	
Predict NLOS	True-NLOS	False-NLOS	NLOS-Precision
Predict LOS	False-LOS	True-LOS	LOS-Precision
	NLOS-Recall	LOS-Recall	Accuracy

ROC-curve

When the learner outputs scores or probabilities, we can evaluate the performance of classification over different thresholds. If say, the threshold to classify a signal as LOS is 0, then all signals are classified as LOS. Put differently, the True LOS rate (LOS-recall) is 1 and

the False LOS rate (LOS-Fall-out) is 1. If the threshold is 1, then no signal is classified as LOS, the True LOS rate is 0 and the False LOS rate is 0.

A ROC-curve visualises the performance of the learner based on True LOS rate and False LOS rate over different thresholds.



The performance of a classifier could then be expressed by the **Area Under the Curve (AUC)**. The AUC takes a value between 0 and 1, where 1 indicates a perfect classifier. In that case, True LOS rate = 1 and False LOS rate = 0, indicating 0 False-LOS and 0 False-NLOS classifications. ROC-curves also provide an efficient way to compare different classifiers.

The optimal operating point of the ROC curve (and the corresponding threshold) can then be determined based on the cost of misclassification, which can be asymmetric if desired.

4.2 Results

The different learning algorithms which were implemented and tested are:

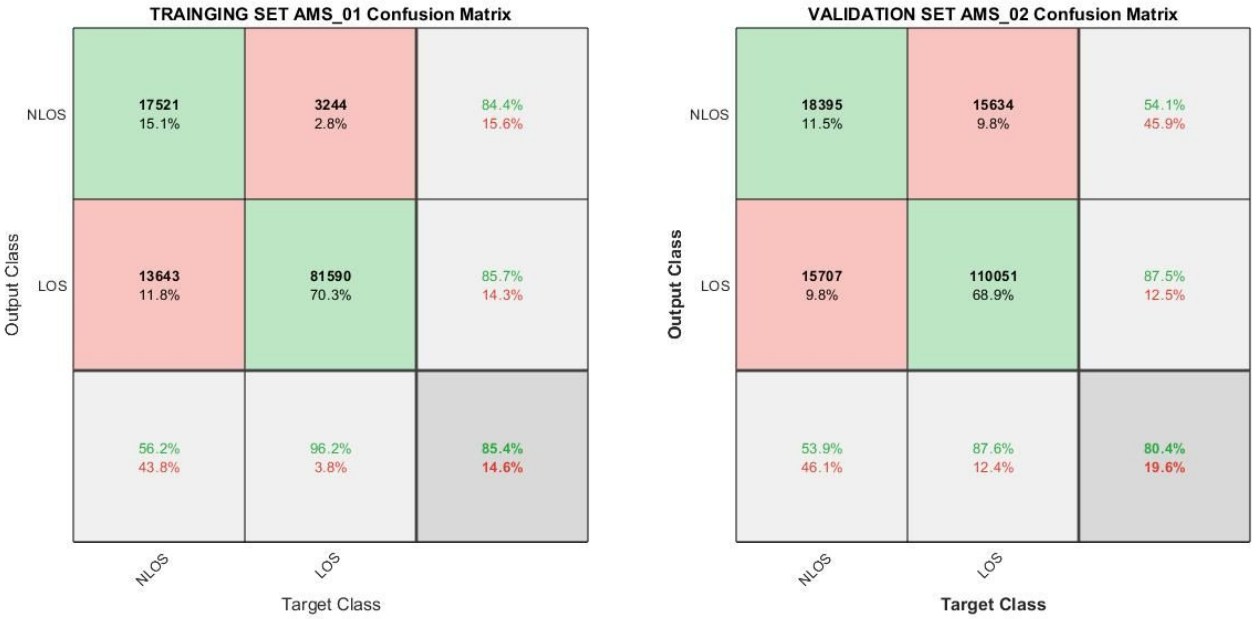
- Decision Tree
- Ensemble of decision trees
- K Nearest Neighbours
- Support Vector Machines
- Neural networks
- Convolutional Neural Networks

I will however not discuss all of them.

Model 1: Decision Tree

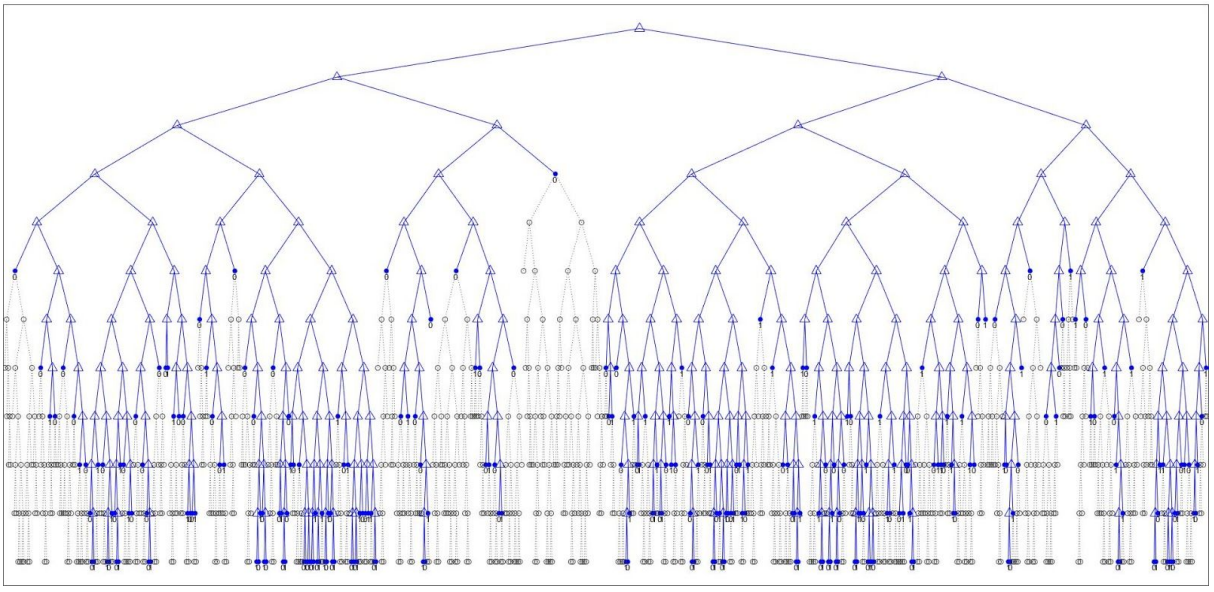
Data	Training: AMS_01 (GPS, GAL) Validation: AMS_02 (GPS, GAL) All data was normalised per constellation.
Inputs	Pseudorange, CNR, Doppler, LLI, Elevation, Third Order Difference, KF Innovation.
Model	**Decision Tree** Maximum number of splits = 800 Minimum number of training data points in a leaf = 1 Split Criterion: Gini's diversity index
Output	Probabilities or Hard classification

For our first model, we will train a decision tree. During training, the tree is constructed by selecting an input variable and finding an optimal split value. When the subset of training data left after following a path of decisions all belong to the same class, the branch is split no further. However, we can also decide to stop splitting after a certain depth is reached. An important (yet difficult) optimisation step is to determine the maximum depth of the tree. Most tree learning algorithms operate in two stages. In the first stage, the tree is grown until a certain depth. In a second stage, the tree is pruned to reduce the size of tree while losing as little accuracy as possible. Another developer decision involves the split criterion (metric to determine which input is best to split on next). We will consider the unpruned base model first.

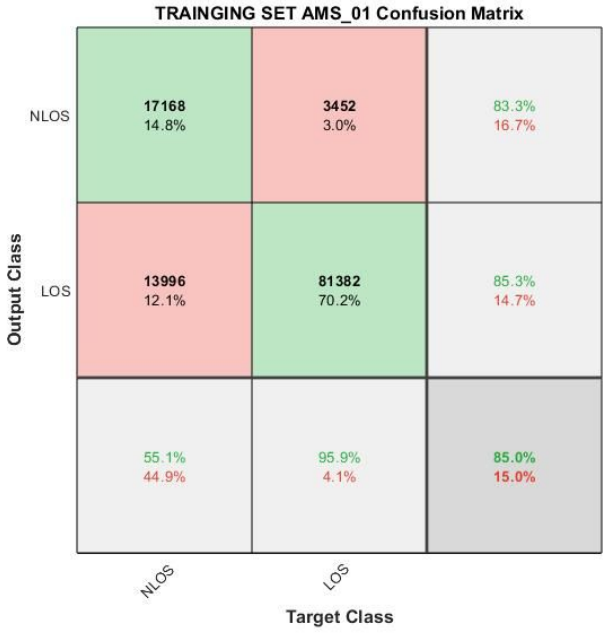


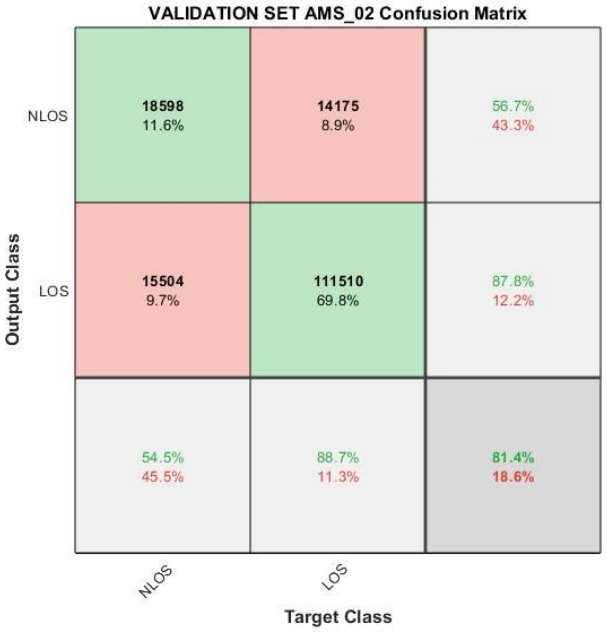
We observe that the decision tree is capable of understanding the training data, obtaining an overall accuracy of 85.4%. However, the learner is performing not as well on data of the AMS_02 tour, which it has never seen during training. The predominant reasons are the NLOS-recall (only 53.9% of NLOS signals are identified) and the NLOS-precision (of all those classified as NLOS, only 54.1% are in fact so). We can conclude that, in order to optimise the overall accuracy, the learner is strongly biased to classify signals as LOS. This behaviour is expected as the overall number of LOS signals (~84000) in the training set is a lot larger than the number of NLOS signals (~31000). The decision tree optimises the total number of classifications. Based on the ROC curve, we could identify a threshold which would bias the learner more towards NLOS signals, at the cost of losing accuracy in general.

As explained before, we can prune the base model to reduce its complexity and stimulate the learner to generalize. The difference between the base case and the pruned tree is shown below (with pruned branches indicated in black).



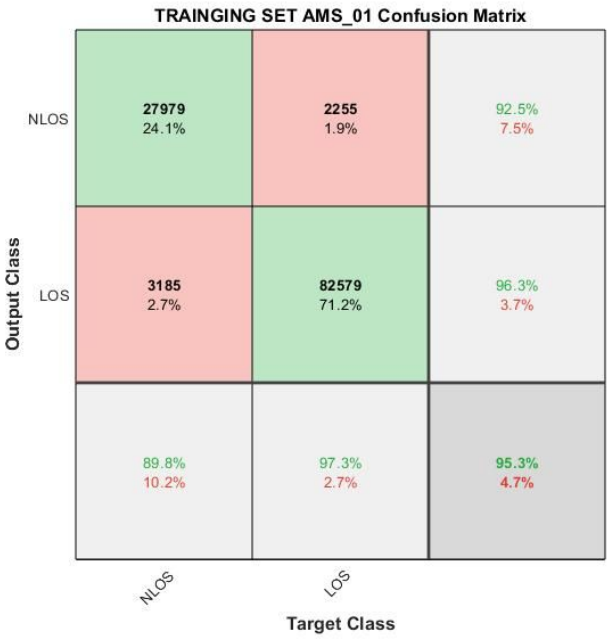
The performance of the pruned tree is shown in the next two figures.

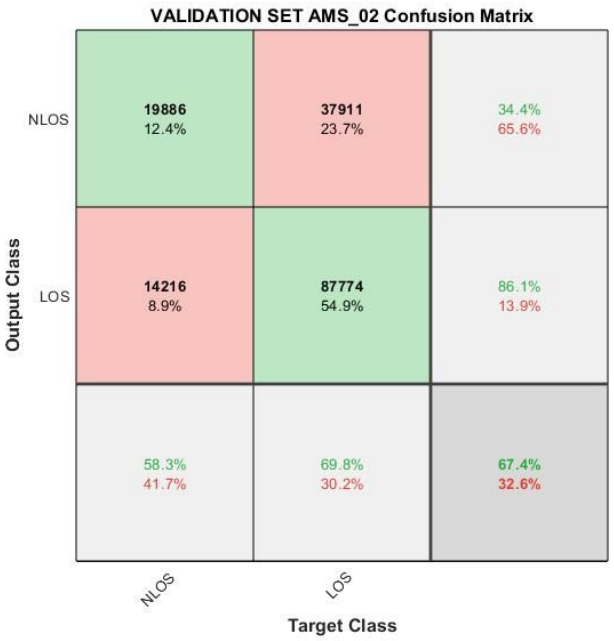




The accuracy on the training set dropped only a small amount, from 85.4% to 85.0%. However, due to the fact that the pruned tree is less fitted on the training data and therefore more general, we see an increase in performance on the validation set, from 80.4% to 81.4%.

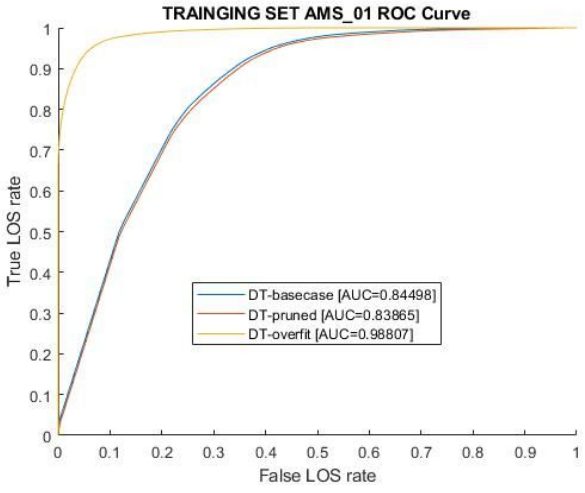
The behaviour of a learner to perform extremely well on training data but poorly on new data is called **overfitting**. To illustrate in how far a decision tree is able to overfit, a decision is trained with maximum number of splits = 10000.

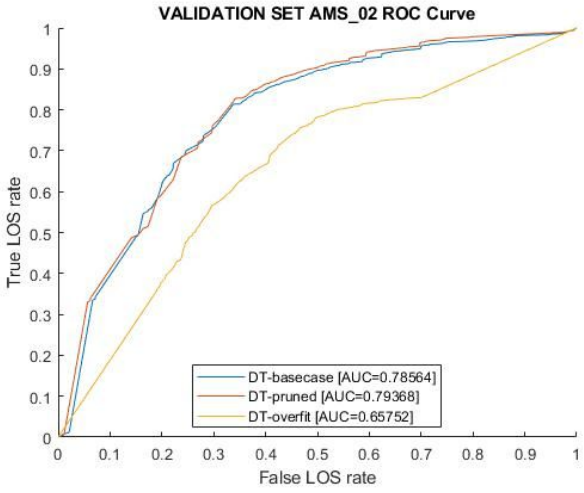




Although marvelous results are found on the training set, the learner's accuracy on the validation set dropped to 67.4%. Notice in particular the large NLOS-precision difference between training and validation, clearly indicating the learner has memorised the training data in extreme detail (or in other words, it learned the noise). However, the learner is unable to translate these patterns to new data.

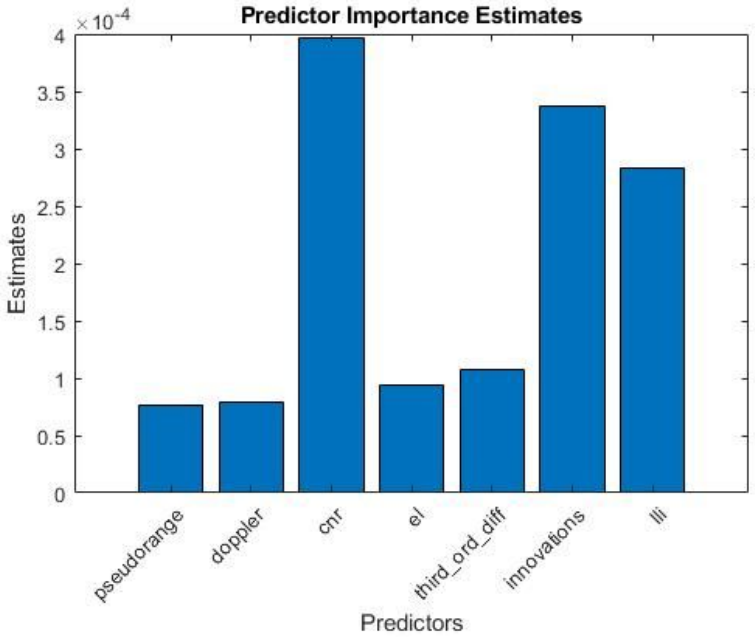
ROC-curves can help us verify our conclusions:





The AUC shows that the pruned Decision Tree (DT) is performing best on the validation set.

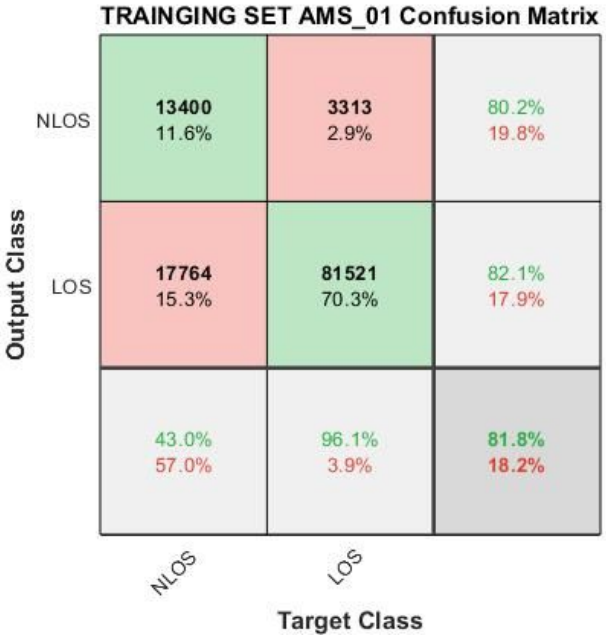
Lastly, MATLAB allows us to calculate the importance of the predictors based on their contribution in reducing the loss function. This therefore provides us insight into what information is most valuable to the learner, as well as what information is valuable to our goal of identifying LOS/NLOS signals.

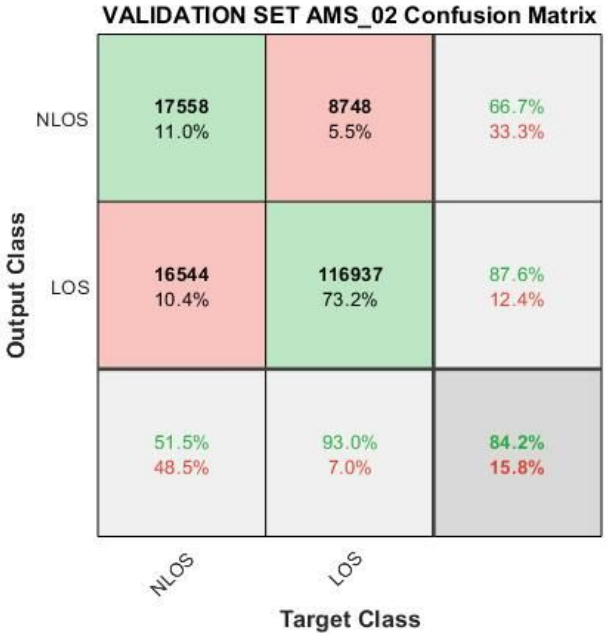


Model 2: Ensemble of Decision Trees

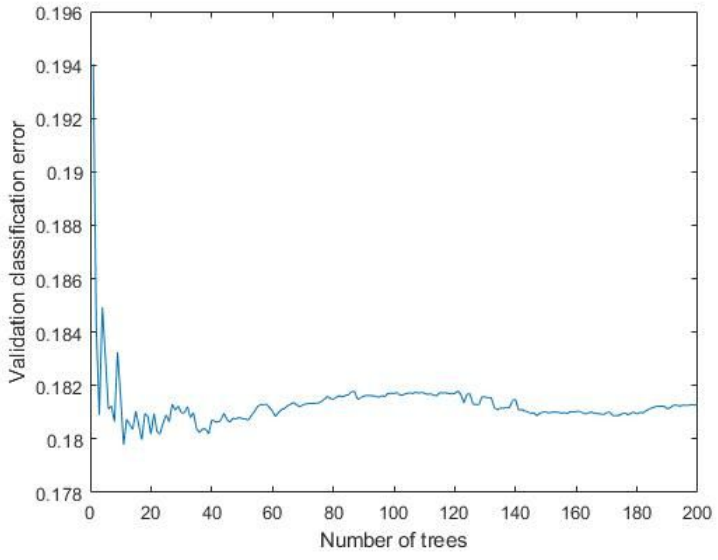
Data	Training: AMS_01 (GPS, GAL) Validation: AMS_02 (GPS, GAL) All data was normalised per constellation.
Inputs	Pseudorange, CNR, Doppler, LLI, Elevation, Third Order Difference, KF Innovation.
Model	**Ensemble:** Ensemble size = 200 Method: Bag or RUSBoost **Base learner (Decision Tree):** Maximum number of splits = 20 Minimum number of training data points in a leaf = 1 Split Criterion: Gini's diversity index
Output	Probabilities or Hard classification

Ensembles combine weak learners and combines their votes into a final decision. Note that the maximum number of splits here is only 20, indicating that each tree in the ensemble is very shallow compared to our previous decision tree model. There are a number of ways to combine trees within an ensemble, which is known as the ensemble aggregation method. The following figures are the results of the *Bagging* method.



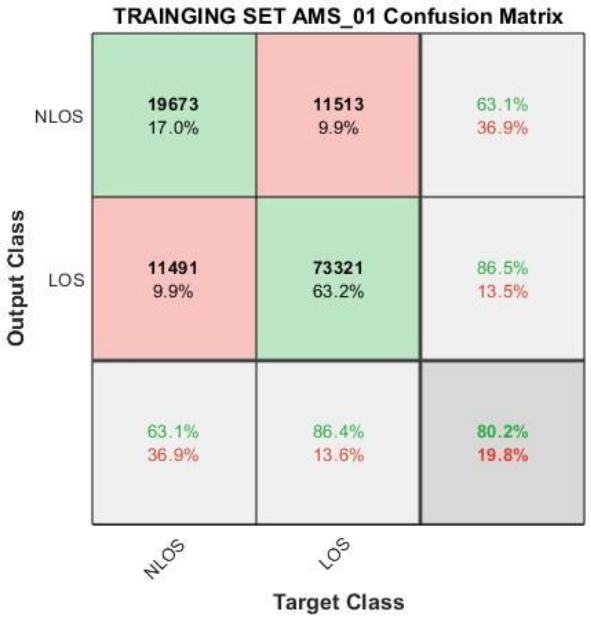


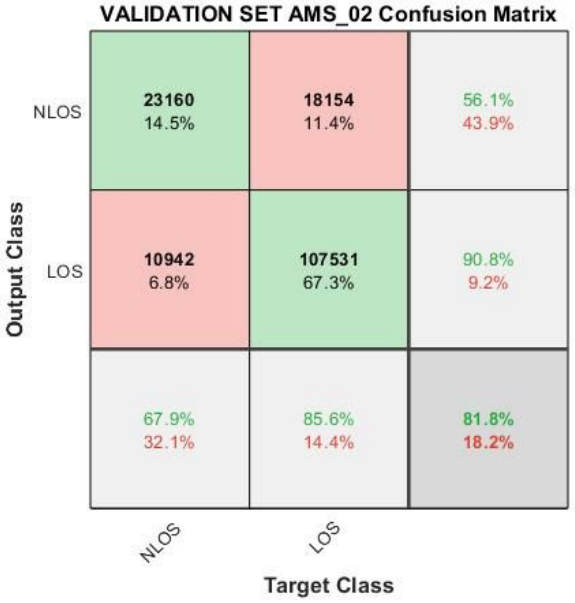
Looking at the training set, the ensemble is doing worse compared to our single decision tree model. However, looking over at the performance on the validation set, we see a strong improvement. We could therefore conclude that an ensemble of weak learners generalises better than one strong learner. We can also visualise the effect of adding more trees to our ensemble against the classification error.



As results are converging quickly, we could decide to reduce the number of trees in the ensemble.

We evaluate the *RUSBoost* algorithm next to help us understand the impact of the aggregation method.





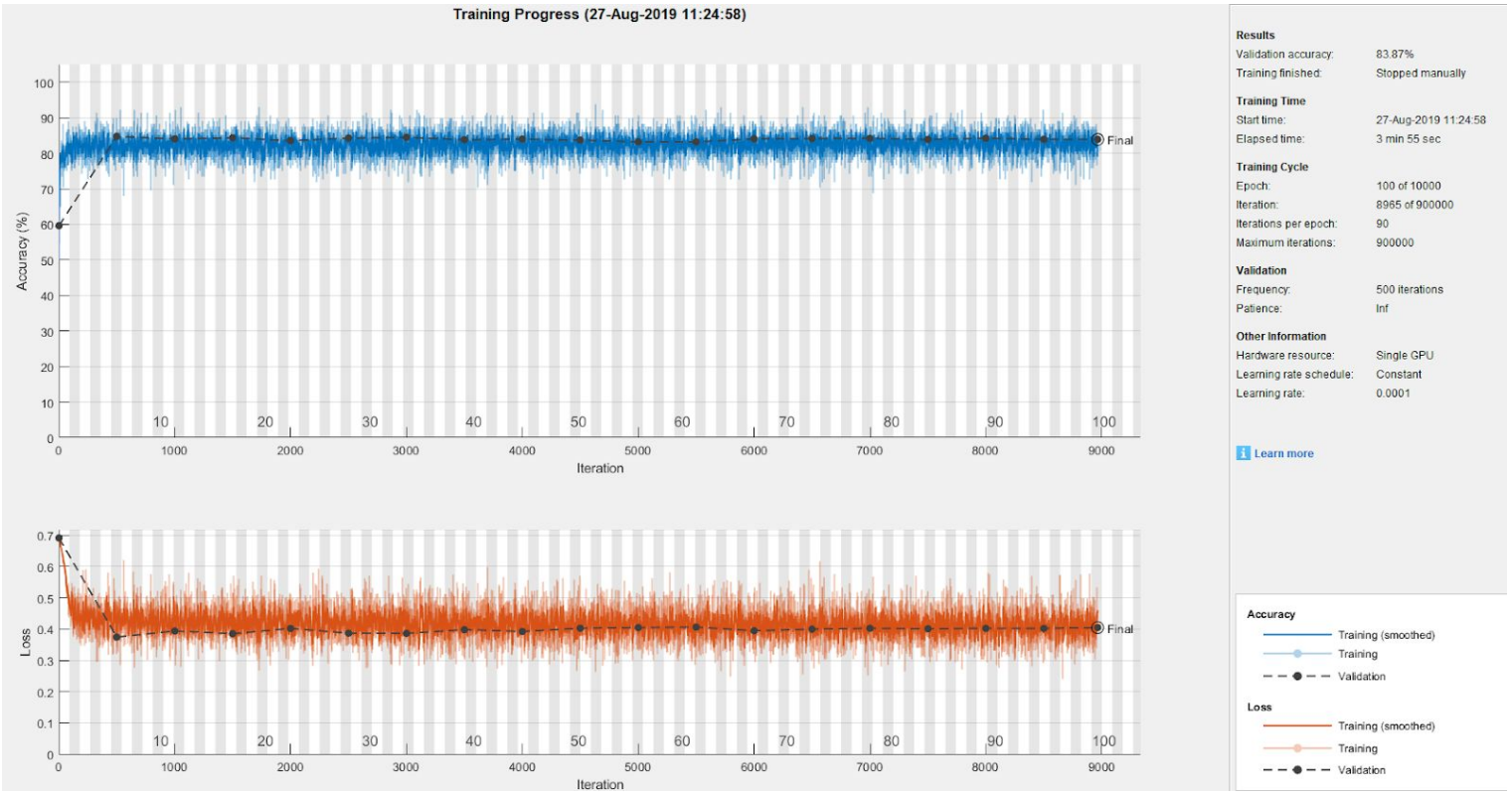
Although the overall accuracy is worse, the algorithm in fact succeeds in obtaining a better balance between the recall over both classes, leading to a decrease in NLOS-precision. Different aggregation methods will optimise different performance measures and choosing the right one is mainly problem-specific. If, for example, we are more interested in a model that is precise when it classifies a satellite as LOS, then the second version is better.

Model 3: Feedforward Neural Network

Data	Training: AMS_01 (GPS, GAL) Validation: AMS_02 (GPS, GAL) All data was normalised per constellation.
Inputs	Pseudorange, CNR, Doppler, LLI, Elevation, Third Order Difference, KF Innovation.
Model	**Feedforward Neural Network:** Inputlayer fullyConnectedLayer (2048 neurons) reluLayer fullyConnectedLayer (1024 neurons)

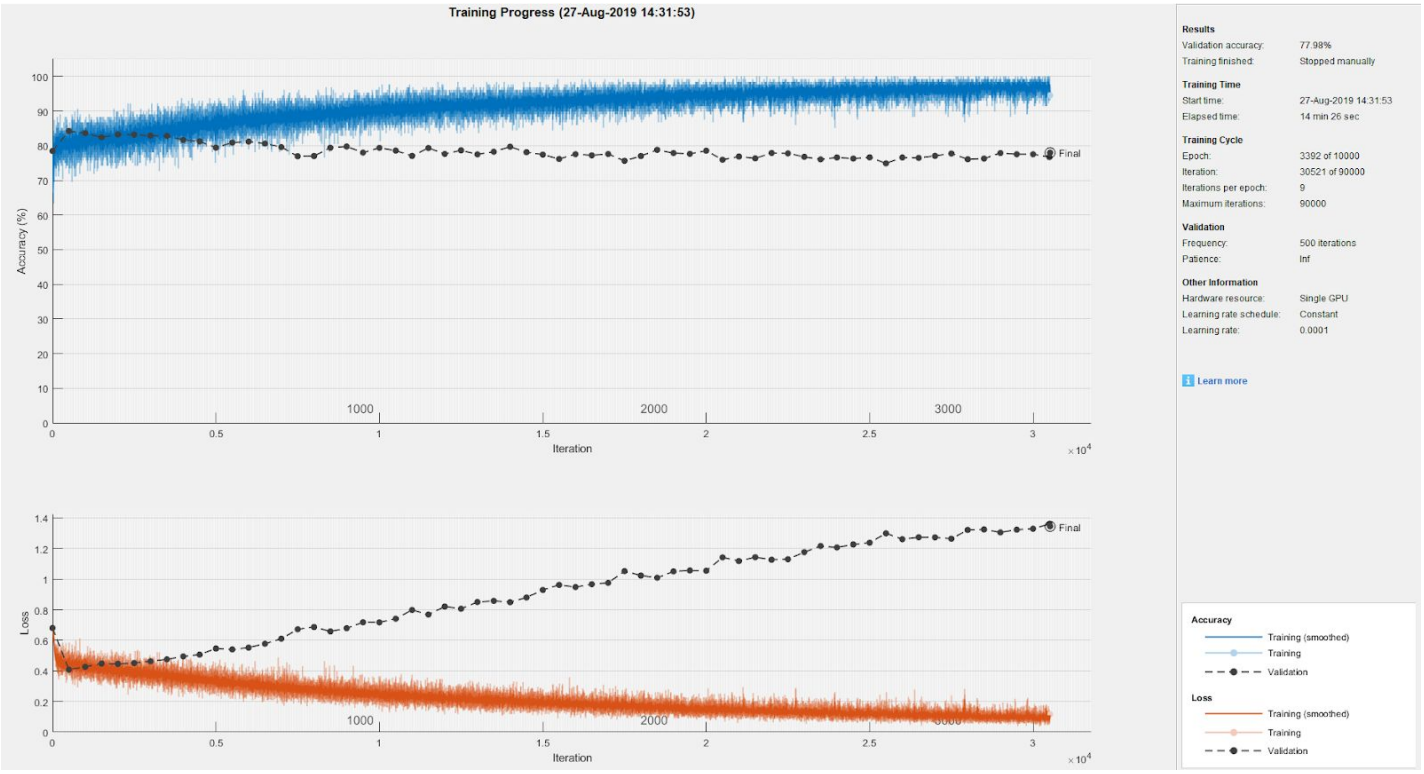
	reluLayer fullyConnectedLayer (512 neurons) reluLayer fullyConnectedLayer (256 neurons) reluLayer fullyConnectedLayer (48 neurons) reluLayer fullyConnectedLayer (2 neurons) reluLayer softmaxLayer classificationLayer
Output	Probabilities or Hard classification

Although I build a relatively simple network, training it on the full AMS_01 dataset is slow. Therefore both the training set is sampled down to 10% of the original dataset. The training result is shown in the following figure.



We observe that the validation closely follows the training, which indicates the learned patterns are in fact valid ones. However, the model seems unable to push the training data to 100% accuracy and therefore seems unable to overfit. This can be considered problematic and can have three causes. First, it may be that the model is simply not complex enough and cannot represent the required patterns (similar to a single neuron which can only be used as a linear classifier). Second, it may be that the data is ambiguous, therefore the learner is simple unable to learn as there is nothing more to learn. Third, it may be that

the hardware we use is inadequate and it is impossible to see long term progress over the first 100 epochs. To confirm that the data and labels are at least to some extent reasonable, we will further reduce the training set to 1% of the original dataset.



Our model is now able to fully overfit on the given training set (but only after 3000 epochs. Reaching 3000 epochs would have taken at least two hours using the previous 5% dataset). Making conclusions on the basis of this test is difficult. We cannot argue that the data is not ambiguous as we are only using a very small subset, and likewise it is hard to conclude that the model is insufficiently complex, as simply more time may be required to learn the required patterns over the entire dataset. We could only test whether the model is complex enough by expanding the current model with more neurons and more layers. However, this will only increase the required training time.

The overall conclusion here is that training these models is in itself a complex process where multiple factors can contribute to lacking validation accuracy. Right now, the overfitting character can largely be explained by the small size of the training set. The best validation possible with this training set is ~85%. As the accuracy on the training set increases, it decreases on the validation set, meaning everything it's learning from that point onwards is in fact noise. In other words, the model is learning the intricate details of the training examples which are not part of the overall pattern. As a result, the loss is actually increasing on the validation set and the accuracy decreasing.

In the long run, obtaining better hardware should alleviate us from one problem. However the question will always remain, *how much gpu power is enough?* So even though more and better hardware can only increase our capabilities, it is in this respect the hardest to validate.

On the other hand, we should more easily be able to verify whether the model complexity or data are in fact not obstructing our performance.

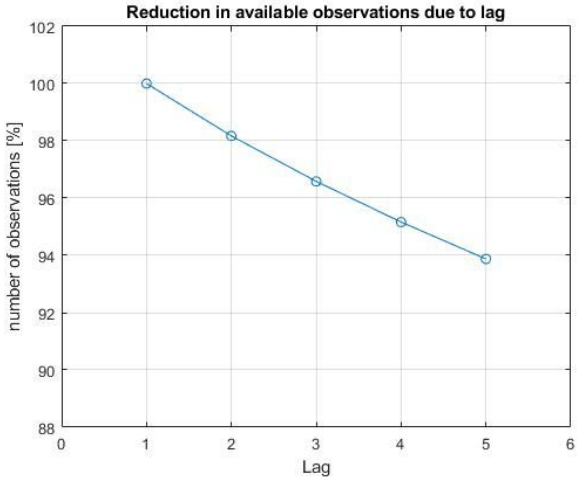
Model 3: Convolutional Neural Network

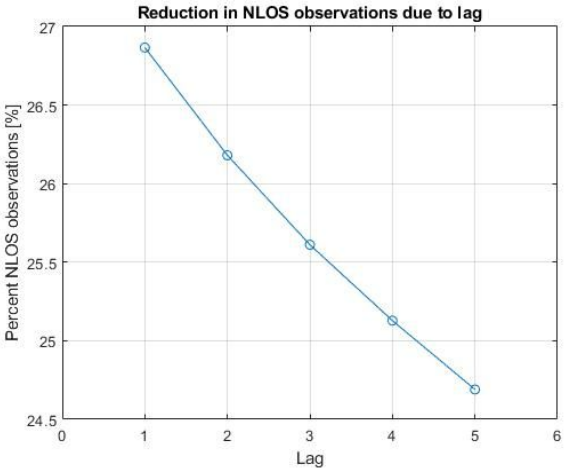
Data	Training: AMS_01 (GPS, GAL) Validation: AMS_02 (GPS, GAL) All data was normalised per constellation.
Inputs	Pseudorange, CNR, Elevation, KF Innovation.
Model	**Convolutional Neural Network** **Layers:** imageInputLayer convolution2dLayer reluLayer dropoutLayer convolution2dLayer reluLayer dropoutLayer maxPooling2dLayer convolution2dLayer reluLayer fullyConnectedLayer reluLayer softmaxLayer WeightedClassificationLayer **Options:** Adam optimizer Initial Learning rate = 0.0001
Output	Probabilities or Hard classification

To investigate our model complexity further, we can attempt a slightly different model. Deep convolutional neural networks (CNN) are the main reason behind the deep learning hype we see today. They are mostly deployed for computer vision purposes, but can be adapted to fit other purposes. The main idea behind a CNN as opposed to a normal neural network is that the layers are not fully connected but only locally. Therefore the number of parameters to train can be reduced significantly, enabling deeper networks which focus on learning relations between nearby features.

For LOS/NLOS classification, we can adapt our input to not only provide the current observed parameters, but also the previous k ones. For simplicity, we will fix the lag hyperparameter k to 5. In other words, the network is given all observable data from a satellite over the past 5 seconds. We therefore can only provide the network with a valid

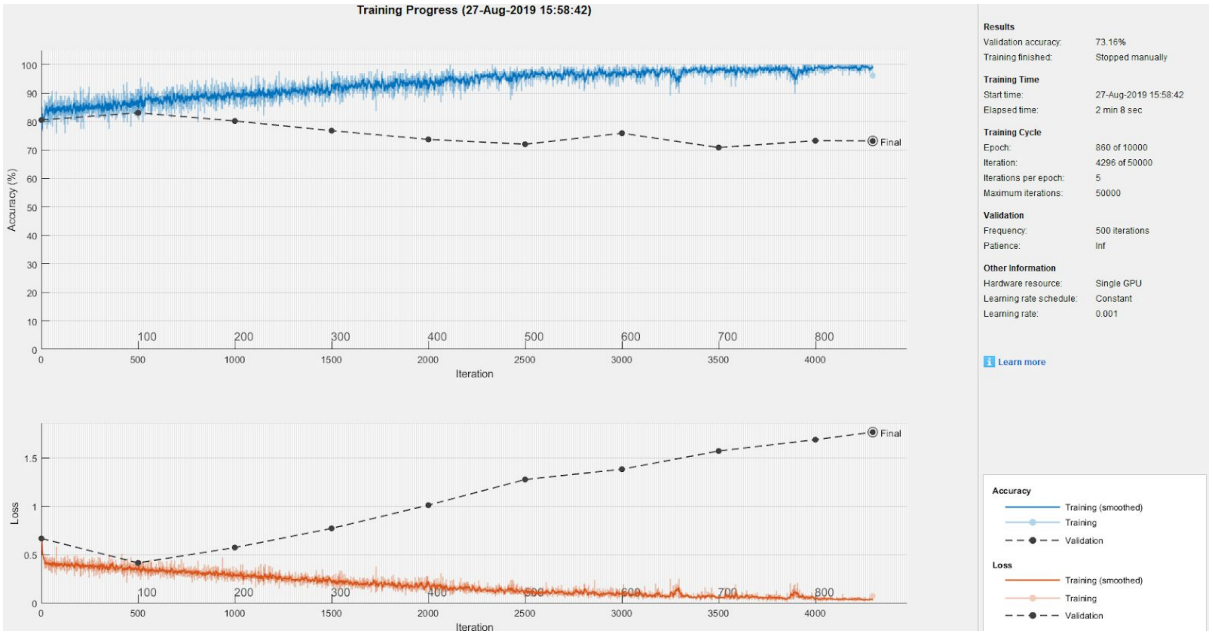
input if the satellite has been observed for 5 consecutive moments. This assumption brings an inevitable cost of losing all data for satellites which have not been observed sufficiently long. We evaluate the effect of the assumption on the dataset in the following figures.



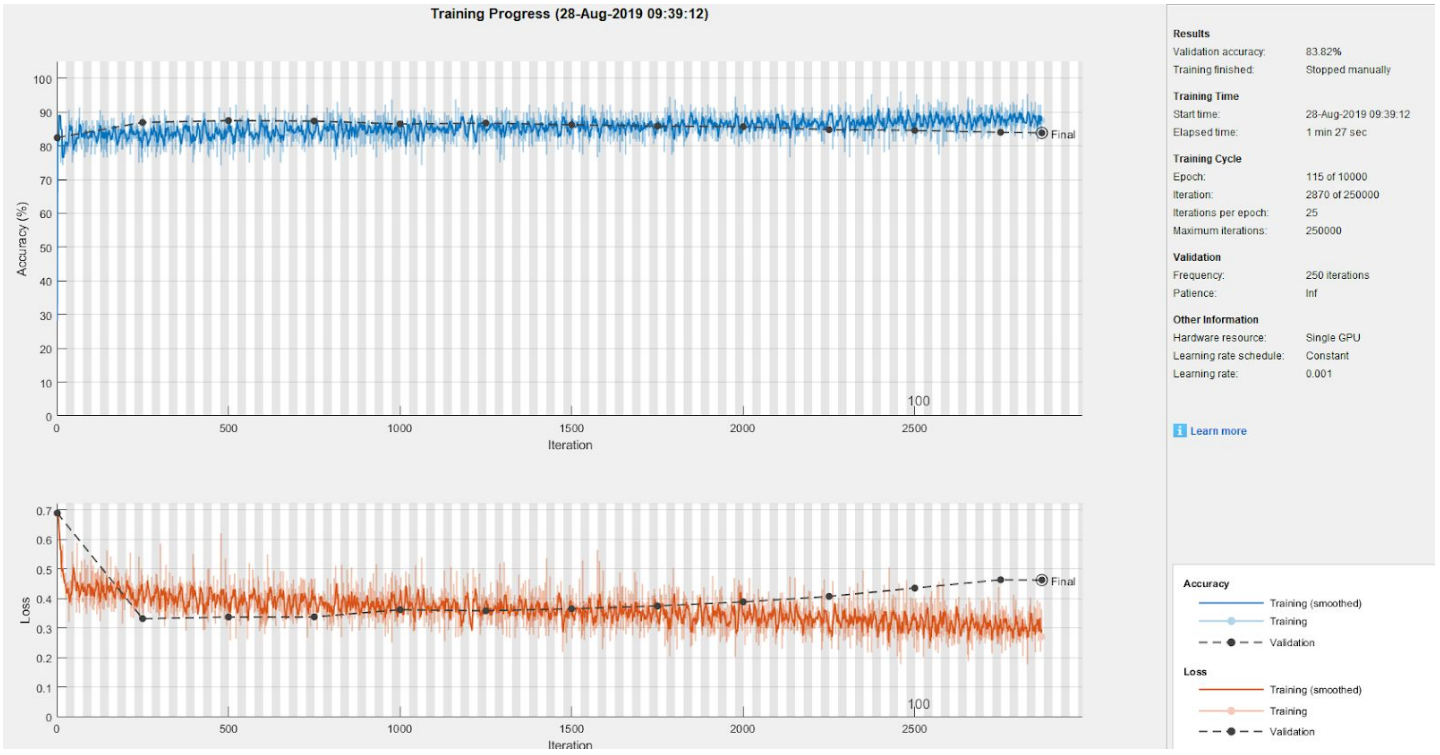


The first figure shows the effect of losing data. The second figure shows the effect on the fraction of NLOS labels. As it turns out, there is a slight correlation between satellites which haven't been observed for longer than 5 seconds and whether or not they are NLOS. In other words, we are losing more NLOS observations than LOS observations. This suggests it might also be interesting to provide the network with the duration of consecutive observations from a satellite. I will however not consider this yet.

To verify our model capability to grasp the main concept, we start off using 1% of the dataset.

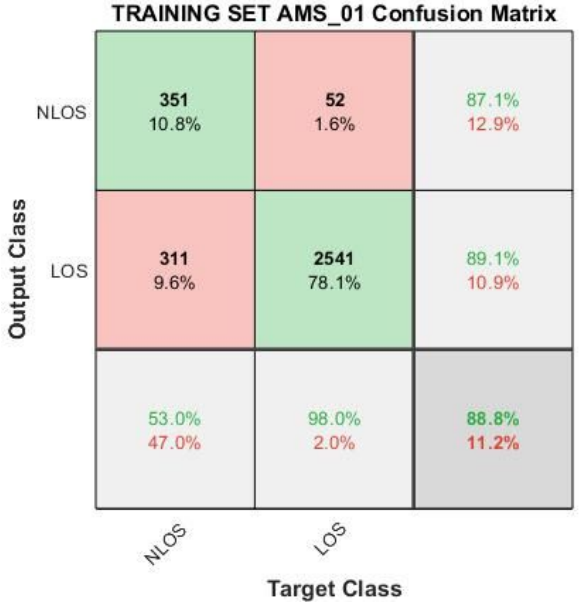


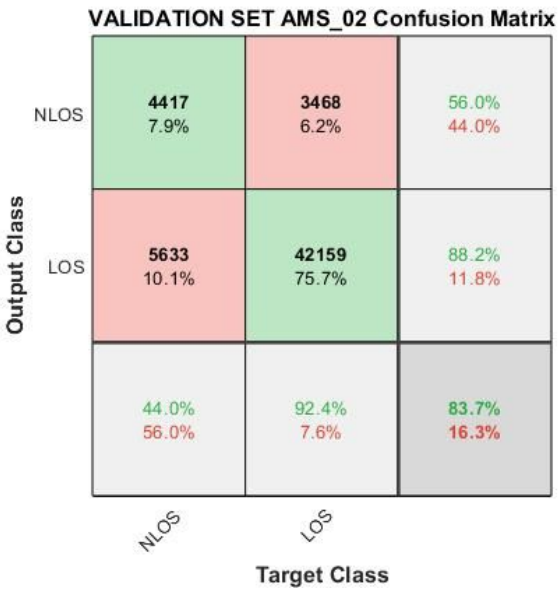
In merely two minutes, the model is able to reach close to 100% accuracy on the training set. However, the validation accuracy peaked near 82% and then dropped to a little above 70%. To strengthen the model, we can move to a training set of 10% of the original data.



Important to see here is that the validation set accuracy is peaking higher compared to using the 1% training set. However, training time is already increasing and the loss on the validation set is increasing as well, indicating the beginning of overfitting. As this model has a strong capability to understand the training data, we can further improve validation accuracy by adding increasing amounts of data, but also by having a better understanding of the data and select a subset of high quality.

Training and Validation Confusion tables for the last trained model:





To have a stronger indication of the consistency of our CNN, we can split up the validation set in partitions and consider the expected value and standard deviation over these partitions.

	E[Precision]	**STD[Precision]**	**E[Recall]**	**STD[Recall]**	**E[F1]**	**STD[F1]**
LOS	88.2%	0.4%	92.4%	0.3%	90.2%	0.4%
NLOS	56.0%	1.2%	44.0%	1.1%	49.2%	0.01%

This table was generated on the AMS_02 tour split in 10 equal partitions. We observe that the standard deviation over precision and recall is about 1%, which helps us to confirm our CNN is in fact very consistent.

5 Future Work

5.1 Data

Although the datasets obtained from the tours form a good start, there are a number of problems and shortcomings with it, most of which could be alleviated by increasing the size of the available data. One notable problem is that the current tour data is highly correlated in time. Therefore, having three hours of data does not translate to three hours of input data for learners. **In general, increasing the amount of available data will allow us to subsample the data to decorrelate the input while maintaining a good representation of the entire input domain.**

To increase the performance of machine learning algorithms, the following suggestions towards constructing a complete dataset should be considered:

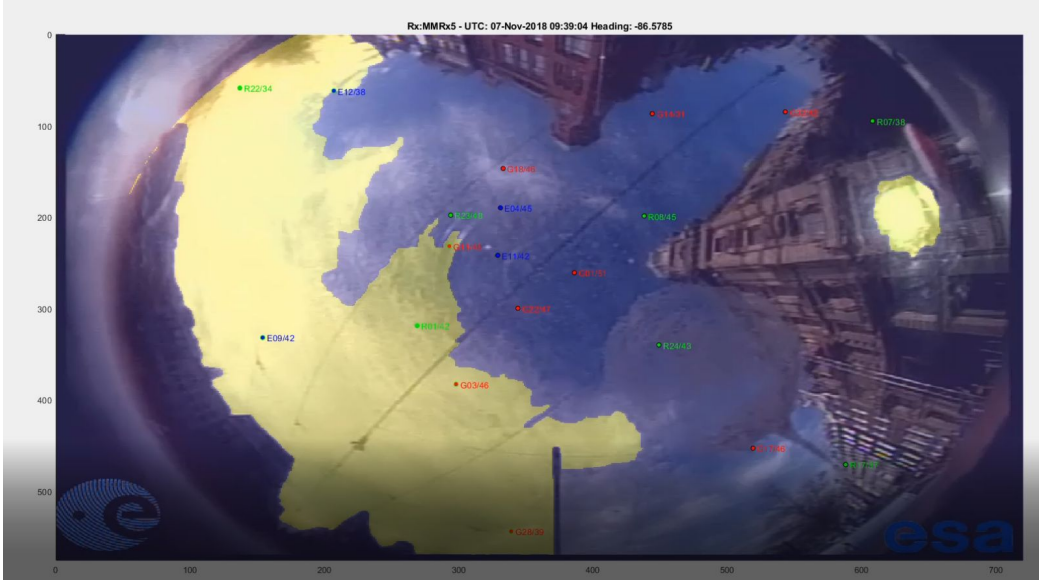
- Data per constellation over 24 hours. The main goal would be to observe the constellation during at least one period (full orbid) of the satellites.
- Data on different locations in urban environments. Different locations will allow us to observe the same constellation over different elevation angles. It should also remove the potential bias factors of one city (e.g. the average street width, nature of reflecting surfaces…)
- Metadata can be helpful. If a certain number of satellites can be grouped together (e.g. if they have a higher CN0 than average), then such information is useful as it helps in identifying different sources of noise.
- Data obtained from different antennas. It might be interesting to experiment and analyse the impact of the antenna on the capability of learning algorithms, which are fundamentally driven by the input which is provided by the antenna and the receiver. We could then observe to what extent having higher precision data helps us in separating LOS from NLOS satellites. Towards this end, simulating a number of different antennas/receivers, from perfect quality to mass market products, should be valuable (if such simulations are possible).

- In line with the last point, perhaps evaluating the additive benefit of dual frequency receivers on learning capabilities can also be interesting.

An ideal clean dataset contains as little bias towards the final objective of classification as possible. In other words, ideally, the **training data** is gathered in a perfectly controlled environment, causing the distribution of input variables not to be influenced by arbitrary circumstances.

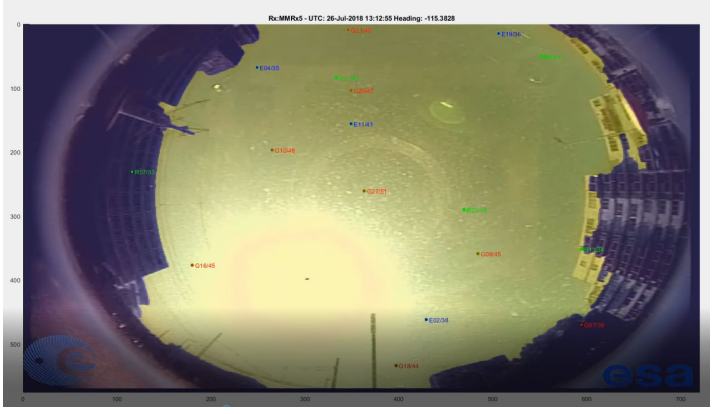
5.2 Labels

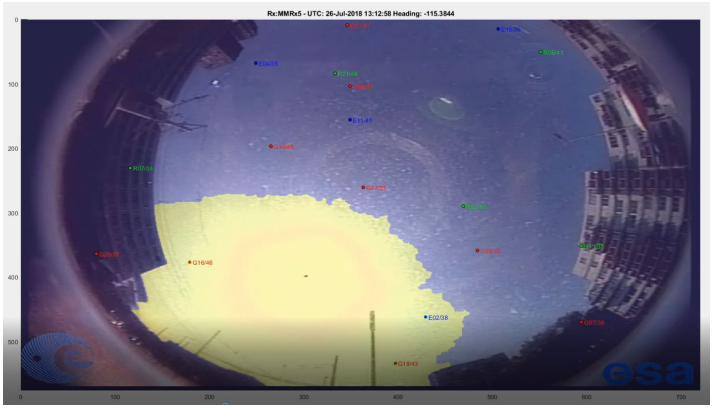
The following image is obtained from the camera segmentation of the AMS_01 tour.



The blue colored background means the algorithm labeled these satellites as NLOS. It is clear that the labelling is therefore at times insufficient.

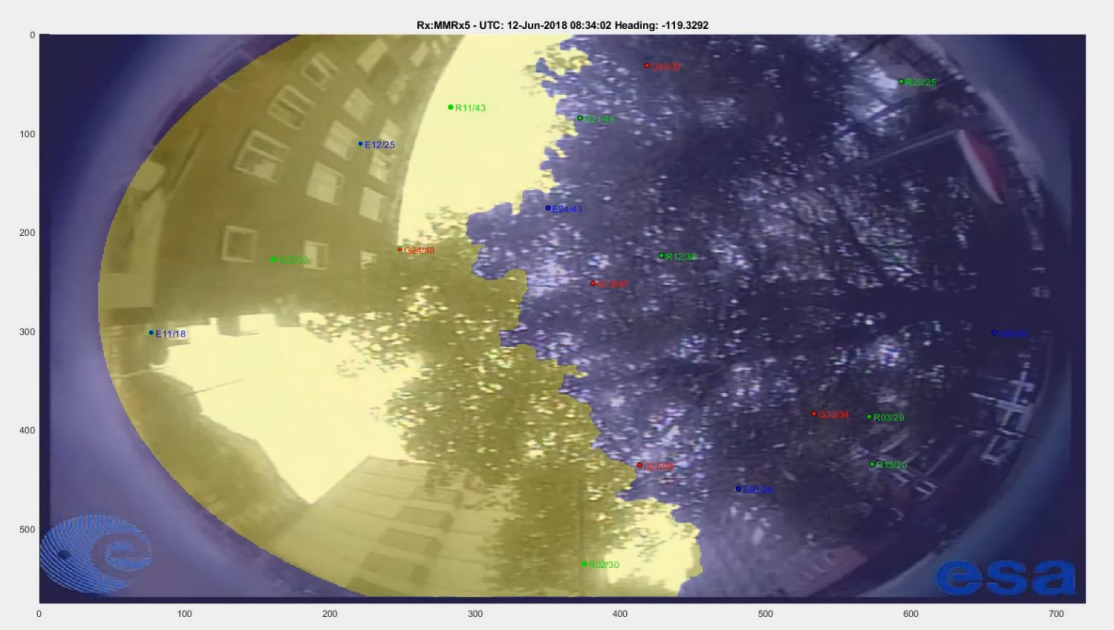
As the image segmentation can change drastically from one image to the next (as illustrated in the following two images), labelling is at times ambiguous, where near-duplicate input information (CN0, elevation,...) is labelled once as LOS and another time as NLOS. Such ambiguity leads to confusion within the learner and can heavily distort the ability to understand which patterns are essential and which ones are noise.





Another problem can emerge when satellites are found on the edge of segmentation regions. In such cases it might be that the satellite is NLOS for the camera, but in fact LOS for the antenna. To illustrate, a couple of borderline cases can be found in the next image.

Vegetation can cause faulty labelling as well, as it is not always clear if the signal in these cases is a reflected one or not. Furthermore, the current labelling algorithm has known difficulties in consistent detection of vegetation as can be observed in the following image.



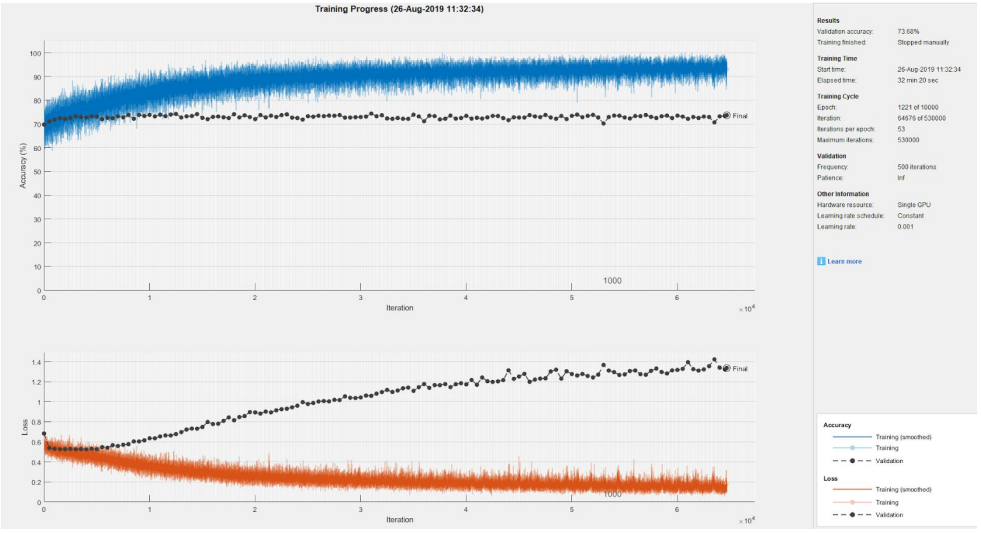
A further problem with the current labelling is that the signal obtained from LOS satellites can still be a reflected multipath signal. Therefore, the data contains both known errors as well as unknown errors.

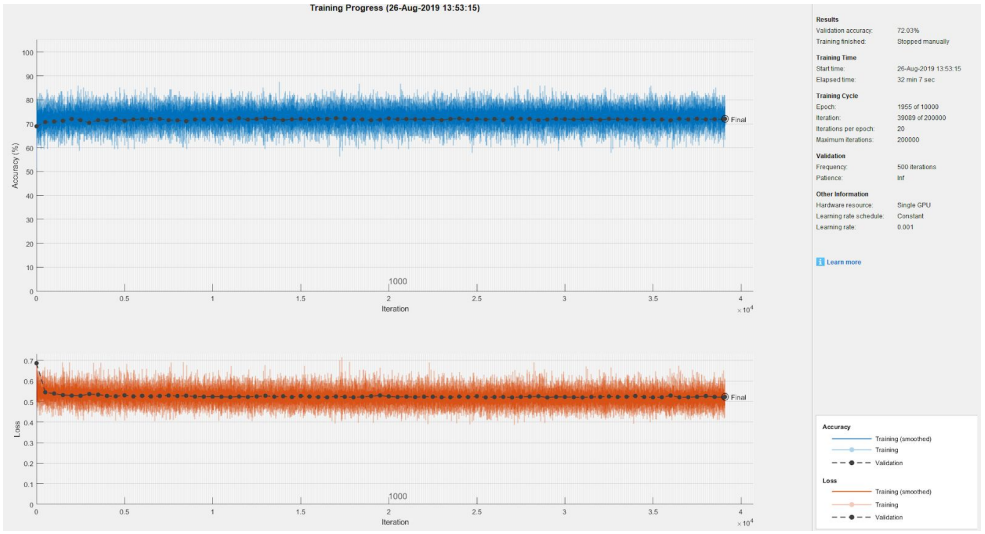
To enable learning algorithms as much as possible, the following label suggestions should be considered:

- Obtaining perfect labelling w.r.t. the camera. Non-perfect labelling can strongly impact the overall performance of learners. The impact will also depend on the nature of the learning algorithm and therefore it will be harder to understand why learners behave a certain way, which is essential in tuning and optimising the algorithms. To obtain perfect labelling, camera footage should be obtained in good segmentation weather (ask Floor, I believe optimal conditions are mostly cloudy), or perhaps with the help of an infrared camera. The objective here is to obtain perfect labelling at all cost. Therefore, we are not concerned with different weather situations and should prefer the best one for image segmentation.
- Ideally, the dataset should contain 50% LOS and 50% NLOS observations. Although the actual urban environment should, at a certain moment in time provide significantly more than 50% LOS satellites, this requirement will help us better analyse our results and better understand the model capabilities towards learning to recognise and distinguish the two classes. E.g.: if a LOS satellite can be identified with more confidence, then this may not be easy to observe when the label distribution is skewed. If the dataset is large enough, we can easily subsample to obtain equal fractions.

5.3 Hardware

Next to expanding the dataset, there are other techniques in deep learning which prevent networks from overfitting, so called regularization techniques. However, applying regularization most commonly requires learners to train a lot more. In combination with more training data, this means the current hardware quickly becomes insufficient. This is illustrated in the next two figures.



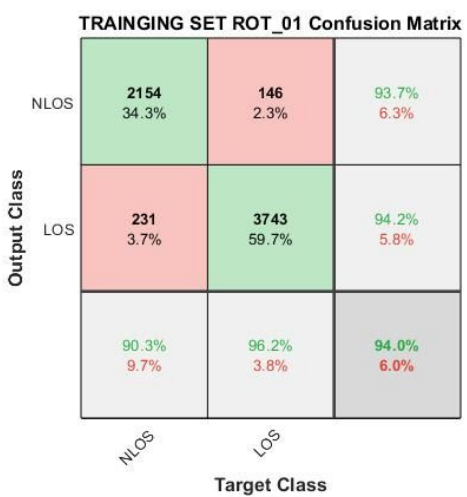


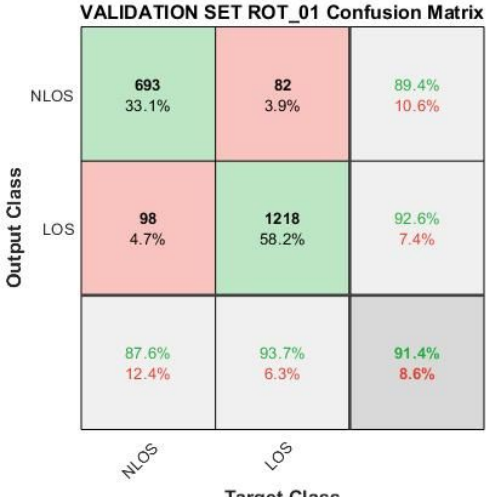
In the first figure, the network is not regularised and clearly overfits on the training data. When regularising the network (figure 2), the network takes an infinite amount of time to increase its accuracy. If the data is clean and the models perform well on small controlled datasets, then upgrading to better hardware will be essential to increase overall performance. This is possible because training networks can be parallelised to a high degree, allowing networks (with billions of parameters) to be trained on massive datasets in hours instead of weeks.

6 A First Look On Future Work...

Current results do not live up to our expectations and I have therefore attempted to identify different aspects which can be improved in the previous section. As we rely heavily on the supervised learning setting, quality of labels is of tremendous importance. First to discover patterns in the data, and second to evaluate the learned patterns.

By observing the camera footage, I have selected a small subset of the ROT_01 dataset which contains minimal vegetation and as little faulty labelling as possible. The subset has a total duration of 14 minutes and 7 seconds, containing 5306 GPS and 3085 Galileo observations. The following confusion tables are produced by training a decision tree learner (and pruning it) on 75% of this dataset, using CNR, elevation and KF innovation as input. The model is then validated on the remaining 25%.





Although this set is too small to have true significance, it can serve as an informal indication of the impact of impure data. The validation results are now closer to the training results and more importantly, the NLOS precision and recall and far better than our previous models (both from training and validation). This may suggest that the identification of NLOS satellites is suffering the most from erroneous labelling.

As the process of labelling satellites based on image segmentation is still an actively ongoing project within ESA, it will be most promising to follow up on their future progress and to return to the LOS/NLOS classification problem using machine learning in due time.