

folder/uio-fp-navn-eng.pdf

teleskopgutt2.jpeg

Master's thesis

A novel application of machine learning to develop pointing models for current and future radio/sub-millimeter telescopes

Bendik Nyheim

Computational Science: Physics
60 ECTS study points

Department of Physics
Faculty of Mathematics and Natural Sciences

Spring 2023

folder/uio-fp

Bendik Nyheim

A novel application of machine
learning to develop pointing
models for current and future
radio/sub-millimeter telescopes

Supervisors:

Signe Riemer Sørensen (Sintef)

Rodrigo Parrar (ESO)

Claudia Cicone (UiO)

Contents

1	Introduction	3
2	Astronomical Background	4
2.1	Astronomy Terms	4
2.1.1	The Celestial Sphere	4
2.1.2	Altitude-Azimuth Coordinate System	4
2.2	Radio/(sub)-mm telescope basics	6
2.3	Problem description	6
2.4	Pointing model	7
2.4.1	Analytical model	7
	Harmonic terms	8
	Az/El non-perpendicularity (NPAE)	8
	Horizontal displacement of Nasmyth rotator	8
	Left-right collimation error	9
	Azimuth and elevation index error	9
	Azimuth axis misalignment	9
2.4.2	Pointing corrections	9
2.5	Research questions and related works	10
2.6	Database	10
2.6.1	Raw Data	10
2.6.2	The Monitor Database	10
	Azimuth and Elevation	11
	Temperature Measurements	11
	Hexapod	12
	Tiltmeter	12
	Weather data	12
	Disp abs?	12
	Automatic adjustments	12
	Data frequency	12
2.6.3	Pointing scan data	14
	Line pointing	14
	Continuum Scan	15
	Pointing scan timestamp	16
	Instruments	17
2.6.4	Tiltmeter dump files	17
3	Machine Learning Background	19
3.1	Supervised learning	19
3.2	Loss/Cost	19
3.2.1	Loss Functions	19

3.3	Train/test set	20
3.4	Scaling	20
3.5	Decision Trees	21
	Bagging	21
	Random Forest	21
	Boosting	21
	Gradient Boosting	22
3.6	Neural Networks	23
3.6.1	Backpropagation	23
3.6.2	Activation functions	24
	Tanh	25
	ReLU	25
3.7	Model Explainability	25
3.7.1	SHAP	26
3.7.2	SAGE	26
4	Method	27
4.1	Cleaning pointing scan data	27
4.1.1	Cleaning rules	27
4.1.2	Pointing scan classifier	27
	Method	27
	Results	28
4.2	Scan duration analysis	28
4.2.1	Method	29
4.2.2	Algorithm	30
4.3	Feature Engineering	31
4.3.1	Different calculations	31
	Median values	31
	Sum of all change	31
	Change since last correction	32
	Max change in time interval	32
	Position of the sun	32
4.3.2	List of features	33
4.4	Transformation of pointing offsets and corrections	33
4.5	Experiments overview	34
4.6	Experiment 1: Pointing Model using Neural Networks	35
4.6.1	Feature Selection	35
4.6.2	Model Architecture	35
4.6.3	Model Evaluation	37
4.7	Experiment 2: Pointing Correction Model	37
4.7.1	Feature Selection	38
4.7.2	Model Architecture	38
4.7.3	Model Evaluation	39
	Bibliography	40

Chapter 1

Introduction

Radio/(sub)-millimeter telescopes are powerful tools used to study the universe at radio, sub millimeter and millimeter wavelengths. These telescopes are designed to capture and detect electromagnetic radiation from space, which can provide valuable insights into a range of astronomical phenomena, from the formation of stars and galaxies to the behavior of black holes. One of the key components of a radio telescope is its reflective surface, which collects and focuses the incoming radiation. Most radio/(sub)-mm telescopes have a large, parabolic dish-shaped primary mirror, which reflects the incoming radiation onto a smaller, secondary mirror. The secondary mirror then reflects the radiation onto a detector or receiver, which records and processes the signals. Unlike optical telescopes, which provide real-time images of what they observe, radio/(sub)-mm telescopes detect and record photons over time, which are then processed to create a composite image or spectrum. However, this process requires highly accurate pointing, as even slight errors in the telescope's orientation can significantly affect the quality of the resulting data. Pointing errors, often referred to as pointing offsets, can be caused by a variety of factors, including thermal deformation of the telescopes components, gravitational deformation, and other environmental factors like humidity and wind. As these factors may change over time, the offsets naturally change as well. To achieve this accuracy, radio/(sub)-mm telescopes use pointing models, which take into account a range of factors that can affect pointing accuracy, including weather conditions, telescope structure, and the position of the target in the sky. The APEX telescope, located in the high-altitude Atacama Desert in Chile, currently uses an analytical pointing model that is effective but still requires regular corrections based on recent observations of pointing offsets. This research aims to investigate the use of sensory data, such as weather data and tiltmeters, to create a more comprehensive pointing model that accounts for the effects of these variables on pointing accuracy. Furthermore, the research will explore the use of machine learning models to replace the analytical model at APEX in its entirety. This would be particularly useful for larger radio/(sub)-mm telescopes like the future AtLAST telescope, where an analytical model will not be available due to the complexity of the telescope's systems. By developing a more advanced and reliable pointing model, this research will contribute to enhancing the capabilities of current and future radio/(sub)-mm telescopes, used to advance our understanding of the universe. **Add in chapter overview when the chapters are fixed**

Chapter 2

Astronomical Background

2.1 Astronomy Terms

2.1.1 The Celestial Sphere

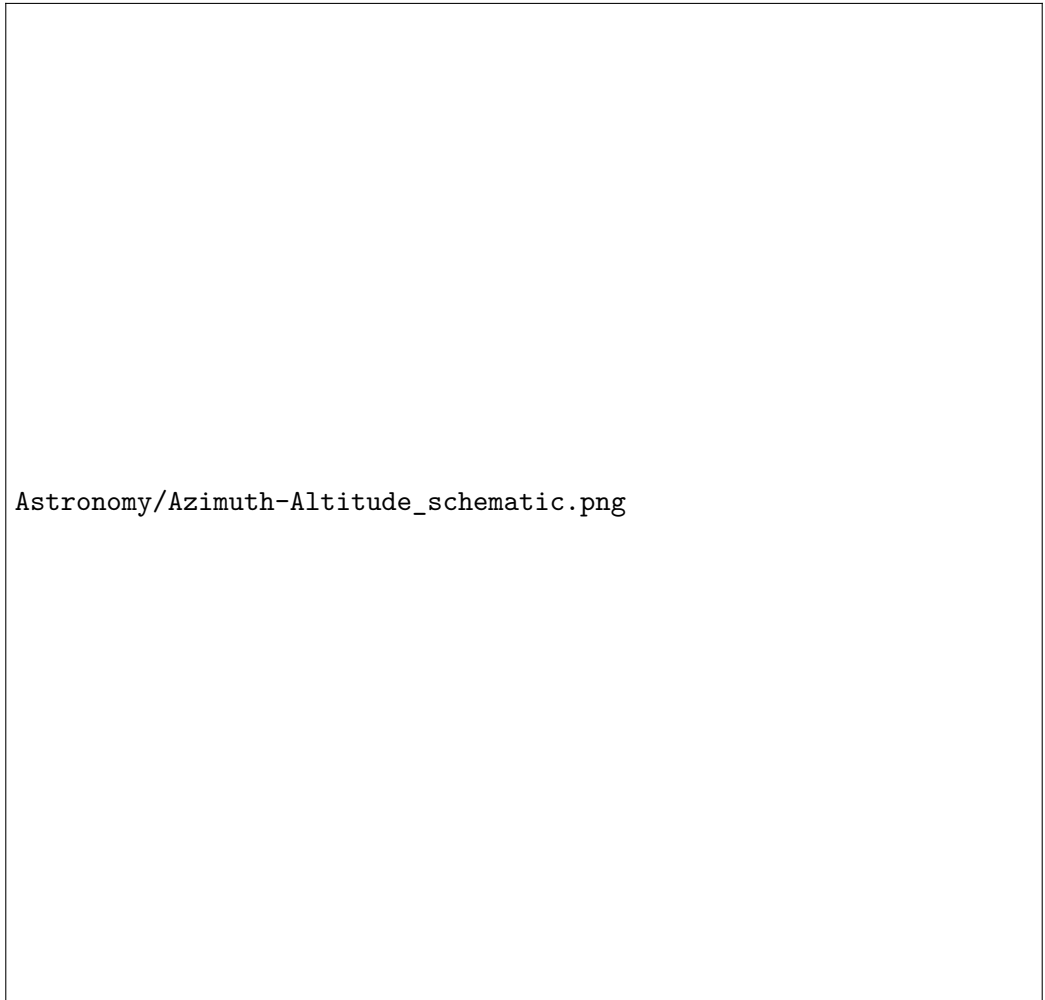
The celestial sphere is a fundamental concept in astronomy. It is an imaginary sphere with an arbitrary radius that is centered on Earth, and it allows us to represent the positions of celestial objects conveniently and intuitively. Any astronomical observation can be thought of as a 2D projection onto the celestial sphere, which is a tool that astronomers use to specify the position of a target without using its distance. Instead, the position is described in terms of two-dimensional coordinates on the sphere. While the celestial sphere is a universal concept, the coordinate system used to specify the location of a target can vary from telescope to telescope.

2.1.2 Altitude-Azimuth Coordinate System

Figure 2.1 depicts the coordinate system used at APEX, which is an altitude-azimuth system. In this system, coordinates are specified using an azimuth and an altitude (or elevation) angle. Azimuth is the angle around the axis perpendicular to the horizontal plane, with zero degrees corresponding to due north. At APEX, the convention is to increase the azimuth angle in a clockwise direction. The interval for azimuth angles is $[-270^\circ, 270^\circ]$ due to APEX's ability to rotate one and a half times around its own axis in the horizontal plane. Elevation, on the other hand, is the angle perpendicular to the horizontal plane, with zero degrees corresponding to the telescope pointing at the horizon, and 90° to the telescope pointing at zenith, which is directly above it. Throughout this thesis, we will use the term elevation instead of altitude to describe this coordinate.

Another angle term used in this thesis is the horizontal angle. We will use the term azimuth when referring to the telescope pointing, while the term horizontal angle will be used for the pointing offset. The azimuth angle is the angle projected on the horizontal plane, while the horizontal angle is the angle measured on the celestial sphere and is dependent on elevation. It is important to be aware of this distinction when measuring offsets, and applying its corrections to the pointing model.

For example, suppose we are pointing at a source at $Az = El = 60^\circ$ and observe that the source is actually 1° to the right. In this case, the horizontal offset is 1° , while the azimuth offset is $1^\circ / \cos El = 2^\circ$. Therefore, the azimuth angle must be increased by twice the horizontal offset due to the influence of elevation.



Astronomy/Azimuth-Altitude_schematic.png

Figure 2.1: The altitude-azimuth coordinate system used at APEX. Source [12]

2.2 Radio/(sub)-mm telescope basics

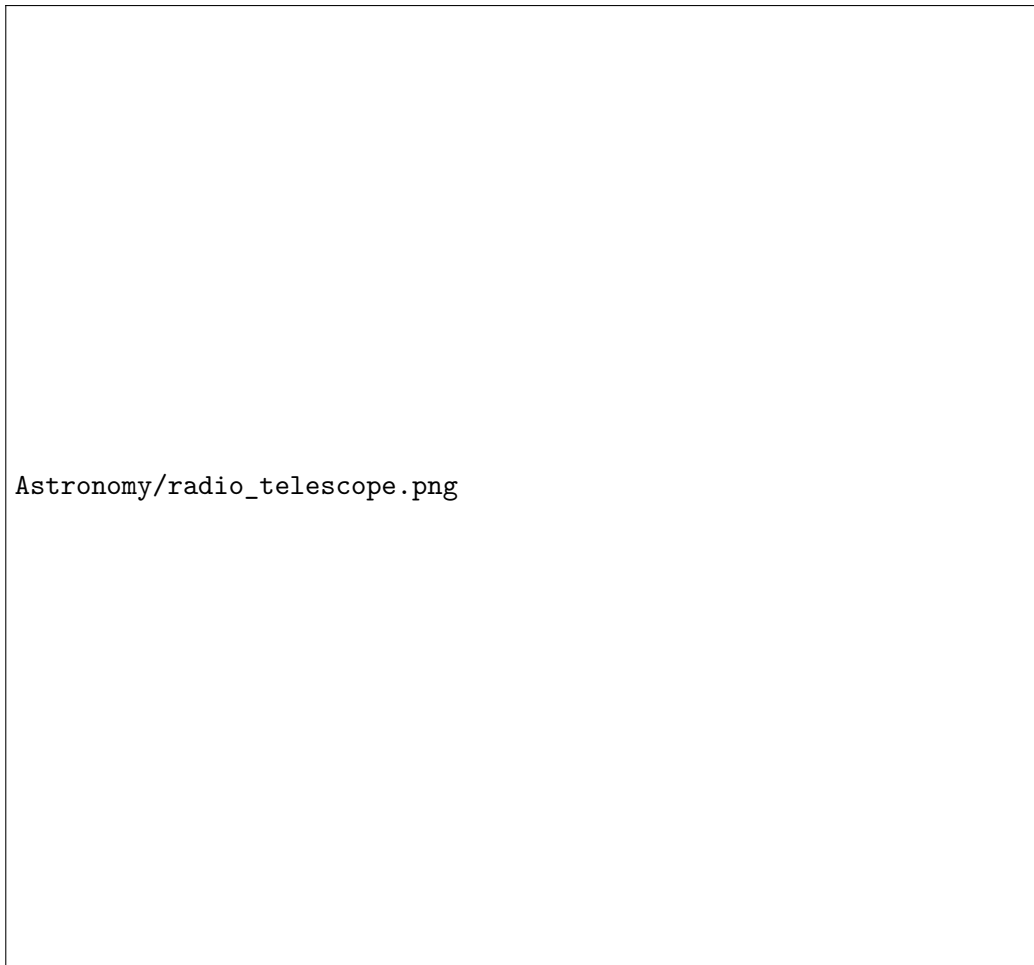


Figure 2.2: The main parts of a radio telescope.

2.3 Problem description

This section have to be fixed completely A telescope makes high precision observations of objects located very far away from earth. Because of this distance, small changes such as deformation of mirrors due to temperature might impact the precision drastically. **Claudia Cicone: Here you could explain that astronomical objects are observed as they were "projected" in 2D angular coordinates on the sky, and you could talk about astronomical coordinates (see Lecture 5 notes of my course AST2210 for some explanation and references)** Over time, these deformations have been observed and analyzed, and in order to counter this, a pointing model is made. **Claudia Cicone: The need for a pointing model is relevant mainly to sub-mm telescopes that cannot take real-time images of what they are observing, and so the precision of the pointing direction must be known accurately before starting the observations.** The pointing model uses measurements from various instruments, and is fitted to the observed offsets. The model is used for about a month, or until it start performing poorly. The variation in measurements are still too big using only this model, so pointing scans have to be made regularly to correct even more. A pointing scan is

an observation of an object with known location. When doing this, the offset on this observation is then added on top of the pointing model. These corrections to azimuth and elevations are used for a couple of hours, until a new pointing scan is made. **Claudia Cicone: Need to define azimuth and elevation first, see other comment above** Using this approach, pointing error is reduced to about 4 arc seconds.

The aim of this thesis is to investigate the possibility of using machine learning to increase the performance of the telescope, by improving pointing accuracy. This can be done in multiple ways.

One possibility is to apply a pointing model on top of the current pointing model using machine learning, such that the average offset is reduced even more. Another possibility is investigating the use of machine learning models to aid such that a pointing scan doesn't have to be made as often. This will reduce the workload at the telescope.

2.4 Pointing model

Since radio/(sub)-mm telescope observes over an extended time, they need a pointing model to obtain sufficiently accurate pointings. The flux of the brightest radio sources is also weaker than the atmospheric emission, which means that they are invisible in real time. Therefore, astronomers need to know that the pointing is accurate before making observations. The pointing model at APEX consists of two parts, one analytical model, and additional pointing corrections performed at regular intervals based on recent observed pointing offsets. The resulting pointing can be explained by these equations

$$Az = Az_{\text{input}} + \Delta Az_{\text{analytical model}} + \Delta Az_{\text{correction}} \quad (2.1)$$

$$El = El_{\text{input}} + \Delta El_{\text{analytical model}} + \Delta El_{\text{correction}} \quad (2.2)$$

Where the first term is the input coordinates, the second is the adjustment made according to the analytical model, and the last term is the adjustments made according to recent observed pointing offsets.

In the following section, I introduce and explain the adjustments from the analytical model and pointing offsets.

2.4.1 Analytical model

How good is this model, and how accurate after pointing corrections? There are a lot of different factors that affect the pointing, and some of them can be modeled with an analytical model. Some of the terms are purely geometrical, based on the imperfect mounting of telescope components, and others are empirical. The following terms are the ones used in the analytical pointing model at APEX, and their description of them are taken directly from the tpoint software manual [13]. All the terms are dependent on either the azimuth Az or elevation El , except for a couple that is just constants. The sum of all the terms is the adjustment made by the model. All the constants are determined by a linear fit based on observed offsets from a pointing campaign.

Harmonic terms

There are multiple harmonic terms in the analytical model, some geometrical and some empirical. The empirical terms have been found to improve the model through trial and error. The following terms are the empirical terms for azimuth

$$\Delta Az = c_1 \cdot \sin Az + c_2 \cdot \frac{\cos 2Az}{\cos El} + c_3 \cdot \cos 3Az + c_4 \cdot \sin 2Az \quad (2.3)$$

$$+ c_5 \cdot \cos 2Az + c_6 \cdot \frac{\cos Az}{\cos El} + c_7 \cdot \frac{\cos 5Az}{\cos El}, \quad (2.4)$$

and the terms for elevation are

$$\Delta El = c_1 \cdot \sin El + c_2 \cdot \cos El + c_3 \cdot \cos 2Az + c_4 \cdot \sin 2Az \quad (2.5)$$

$$+ c_5 \cdot \cos 3Az + c_6 \cdot \sin 3Az + c_7 \cdot \sin 4Az + c_8 \cdot \sin 5Az \quad (2.6)$$

In the tpoint software, harmonic terms are denoted on the format *Hrfci*. The list below explains the different terms.

- *H*: Stands for harmonics
- *r*: The resulting variable, either *Az* or *El*, denoting azimuth and elevation respectively. The resulting variable can also be *S*, which means the result is horizontal, or azimuth scaled by a factor $1/\cos El$.
- *f*: The harmonic function, either *S* or *C* denoting *sine* and *cosine*.
- *c*: The variable that the function *f* is dependent on, either *Az* or *El*.
- *i*: Integer value in the range 0-9, denoting the frequency of the harmonic.

For example will $\Delta Az = HACA3 \cos 3Az$ be denoted as HACA3 in the software.

Az/El non-perpendicularity (NPAE)

In an altazimuth mount, if the azimuth axis and elevation axis are not exactly at right angles, horizontal shifts proportional to $\sin El$ occur. This effect is zero when pointing at the horizon, and increases with elevation proportional to $1/\cos El$

$$\Delta Az \simeq -\text{NPAE} \frac{\sin El}{\cos El} = -\text{NPAE} \tan El, \quad (2.7)$$

where NPAE is the horizontal displacement when pointing at Zenith.

Horizontal displacement of Nasmyth rotator

In a Nasmyth altazimuth mount, a horizontal displacement between the elevation axis of the mount and the rotation axis of the Nasmyth instrument-rotator produces and image shift on the sky with a horizontal component

$$\Delta Az \simeq -\text{NRX}, \quad (2.8)$$

and an elevation component

$$\Delta El \simeq -\text{NRX} \sin El, \quad (2.9)$$

where NRX is the horizontal displacement.

Left-right collimation error

In an altazimuth mount, the collimation error is the non-perpendicularity between the nominated pointing direction and the elevation axis. It produces a horizontal image shift given by

$$\Delta Az \simeq -CA \sec El \quad (2.10)$$

Azimuth and elevation index error

Index errors are the errors when you point at the origo.

The azimuth index error is

$$\Delta Az = -IA, \quad (2.11)$$

and elevation index error is

$$\Delta El = IE \quad (2.12)$$

Azimuth axis misalignment

In an altazimuth mount, misalignment of the azimuth axis north-south or east-west cause errors. The errors cause by misalignment in north-south are given by

$$\Delta Az \simeq -AN \sin Az \cdot \tan El, \quad (2.13)$$

and

$$\Delta El \simeq -AN \cos Az, \quad (2.14)$$

where AN is the misalignment alignent in north-south direction. The errors given by misalignment in east-west are given by

$$\Delta Az \simeq -AW \cos Az \tan El, \quad (2.15)$$

and

$$\Delta El \simeq AW \sin Az, \quad (2.16)$$

where AW is the misalignment alignent in east-west direction.

2.4.2 Pointing corrections

The analytical pointing model can only reduce the pointing offsets down to about an average of x arcseconds. To reduce the pointing offsets even further, a pointing scan is made by pointing the telescope at a source with known coordinates. This operations is called a pointing scan, and by observing the resulting pointing offsets from the known source, the pointing model can be updated according to the following eqautions The terms $\Delta Az_{\text{correction}}$ and $\Delta El_{\text{correction}}$ are updated

$$\Delta Az_{\text{correction}} = \Delta Az_{\text{correction}} + \delta_{Az} \quad (2.17)$$

$$\Delta El_{\text{correction}} = \Delta El_{\text{correction}} - \delta_{El}, \quad (2.18)$$

where δ_{Az} and δ_{El} are the recently observed pointing offsets in azimuth and elevation respectively. These pointing corrections are carried out every couple of hours to make sure the pointing is sufficient during science observations.

In the telescope systems, the correction variables $\Delta Az_{\text{correction}}$ and $\Delta El_{\text{correction}}$ are labeled as *ca* and *ie* respectively, and I will therefore refer to them as such in the rest of this thesis.

2.5 Research questions and related works

- Reduce pointing offsets with machine learning model
- Replace analytical model with machine learning model
- reduce the frequency of pointing scans while maintaining pointing accuracy with machine learning model
- Article about analytical model
- ...

2.6 Database

2.6.1 Raw Data

The analytical model relies on observations obtained through an optical receiver. Optical observations are easier to obtain when there is no other pointing model, as you can observe the source in real time. To obtain the necessary data, the telescope is pointed at various sources with known locations. This process yields both input coordinates and observed coordinates, and can be done for the different instruments at the telescope. According to astronomers at the telescope, only the NFLASH230 receiver have reliable and consistent optical observations, as the data obtained using the other receivers are too noisy due to structural changes at the telescope. Therefore, only optical data from NFLASH230 is used for the analytical underlying pointing model in this project.

Table 2.1 provides an example of the format of this data.

Date	Input		Observed	
	Azimuth	Elevation	Azimuth	Elevation
2022-01-03 14:24:04	189.812879	41.0762	190.254779	40.883651
2022-01-03 18:59:40	50.842145	73.371647	51.269044	73.203243
2022-01-03 19:01:49	49.555916	73.752182	49.983112	73.583545
2022-01-03 19:16:10	39.378382	76.076236	39.781084	75.908956
2022-01-03 19:18:27	113.934309	39.345667	114.391232	39.170168
2022-01-22 13:54:31	94.04365	18.148405	94.492505	17.981161
2022-01-22 14:15:35	148.569964	89.044036	147.783271	88.852306
2022-01-22 14:18:15	215.664924	49.563821	216.104389	49.386438

Table 2.1: Extract from data obtained by the optical receiver of NFLASH230. The source is also included in the datafile, but it is not relevant here. **I don't know what it means that optical data is for the nflash receiver**

2.6.2 The Monitor Database

The monitor database plays a critical role in this project, providing valuable sensory data from within and outside of the telescope. In this section, we will explore the data contained within the monitor database and identify the variables that are most relevant for our purposes.

Azimuth and Elevation The database have tables for the input azimuth and elevation, which is the raw coordinates before the main pointing model have adjusted the pointing. These are labeled COMMANDAZ and COMMANDEL. In addition to these, it includes the actual azimuth and elevation, which is the result after applying the pointing model, and a correction obtained from a pointing scan. These are labeled ACTUALAZ and ACTUALEL.

Additionally, the database contains tables for the velocity of the azimuth and elevation, labeled ACTUALVELOCITYAZ and ACTUALVELOCITYEL.

Temperature Measurements Temperature is measured by multiple instruments at various locations on the telescope. The database includes tables labeled as TEMPERATURE, TEMP1 through TEMP6, TEMP26 through TEMP28, and TILT1T. Figure 2.3 shows that many of these measurements are highly correlated. Specifically, TEMP1 through TEMP6 are all correlated with each other at a coefficient of ≥ 0.98 . Similarly, TEMP26 through TEMP28 and TEMPERATURE are also highly correlated.



Figure 2.3: Linear correlation between temperature measures. The values are the median of all data points within a pointing scan.

Hexapod The secondary mirror, also known as the subreflector, is supported by a hexapod. The hexapod is capable of three-dimensional movement and can also rotate around the azimuth and elevation axes. There are five measures associated with the hexapod: POSITIONX, POSITIONY, POSITIONZ, ROTATIONX, and ROTATIONY. These measures are essential for positioning the secondary mirror and ensuring an accurate pointing.

Tiltmeter The telescope is equipped with two tiltmeters that measure the tilt or inclination of the telescope with respect to the vertical direction. One of the tiltmeters is aligned with the telescope pointing direction, while the other is orthogonal to it. These tiltmeters are labeled TILT1X and TILT1Y, respectively.

Weather data The weather station at the telescope provides measurements of various weather parameters, including dew point, humidity, pressure, wind speed, and wind direction. The instruments take measurements at an interval of 11 seconds.

Put this in feature engineering section

$$\Delta Az_{wind} = Az_{pointing} - Az_{wind} \quad (2.19)$$

For the turbulence, a simple model is used

$$v_{wind} = \sigma_{wind}^2 \quad (2.20)$$

Disp abs?

Automatic adjustments To ensure accurate and stable pointing of the telescope, there are automatic adjustments made based on readings from various sensors. These adjustments account for systematic errors that have been previously modeled and are based on measurements from tiltmeters, temperature sensors installed at different locations, and other relevant sources of data [Explain spem and disp here?](#). The automatic adjustments are made in the azimuth and elevation directions and are denoted by variables starting with DAZ or DEL, respectively. A comprehensive list of these variables can be found in Table 2.2.

Data frequency The monitor database provides data with varying frequencies, as shown in Table 2.2, which lists the approximate number of data points per minute.

Table 2.2: The frequency in datapoints per minute of different variables in the monitor database.

Table	Frequency [datapoints/minute]
ACTUALAZ	6
ACTUALEL	6
ACTUALVELOCITYAZ	6
ACTUALVELOCITYEL	6
COMMANDEL	6
COMMANDAZ	6
TILT1X	12
TILT2Y	12
TILT1T	12
TEMPERATURE	5
TEMP1	6
TEMP2	6
TEMP3	6
TEMP4	6
TEMP5	6
TEMP6	6
TEMP26	2
TEMP27	2
TEMP28	2
DAZ_TEMP	12
DAZ_TILT	12
DAZ_TILTTEMP	12
DAZ_SPEM	12
DAZ_DISP	12
DAZ_TOTAL	12
DEL_TEMP	12
DEL_TILT	12
DEL_TILTTEMP	12
DEL_SPEM	12
DEL_DISP	12
DEL_TOTAL	12
POSITIONX	6
POSITIONY	6
POSITIONZ	6
ROTATIONX	6
ROTATIONY	6
DISP_ABS1	12
DISP_ABS3	12
DISP_ABS2	12
DEWPOINT	5
PRESSURE	5
HUMIDITY	5
WINDSPEED	5
WINDDIRECTION	5



Figure 2.4: Timeseries plots of different sensory data before, during and a little after a pointing scan. The red line denotes the timestamp for a scan in the pointing scan database. The red dots are from when the telescope is observing, while the blue dots are when the telescope is idle or preparing to observe.

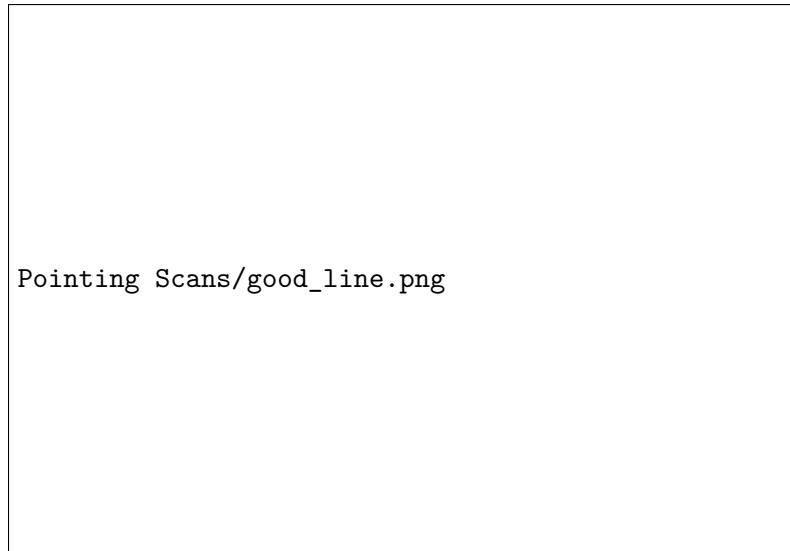
2.6.3 Pointing scan data

A pointing scan is an observation of a source with known location, which is used to calibrate the pointing model. There are two types of pointing scans: Line pointings and continuum scans. Figure 2.4 shows the information in the monitor database form a pointing scan.

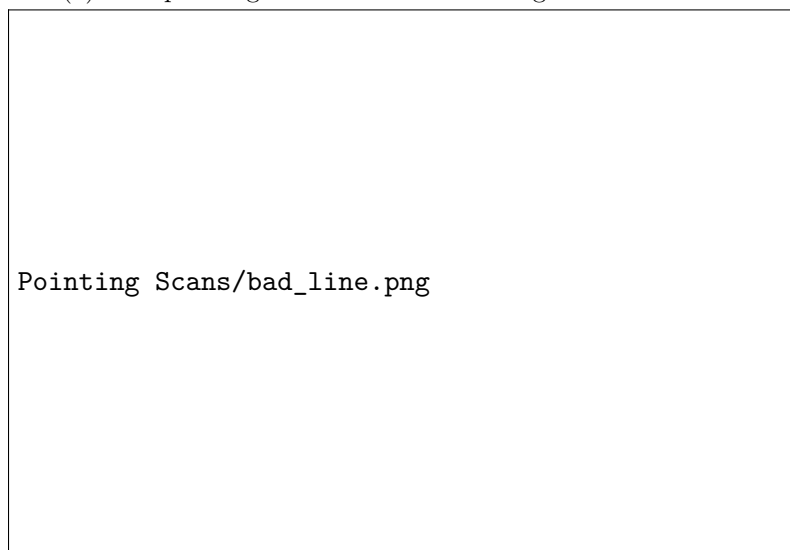
Line pointing A line pointing involves pointing at an extended source. The telescope then makes ten scans, recording the flux intensity from the source, five vertically and five horizontally, around the center of the pointing, as shown in figure 2.5. The upper panel shows a high quality pointing scan and the lower panel shows

a noise low quality pointing scan. The cross-plot on the right side shows the line spectrum for each of the observations (center plus eight offset observations).

The flux from the source is integrated, and the resulting values are plotted as blue dots on the left-hand side of the plots. A Gaussian is then fitted to these points, and the resulting amplitude, full width at half maximum (FWHM), and offsets are given in the table.



(a) Line pointing with little noise and a good Gaussian fit.

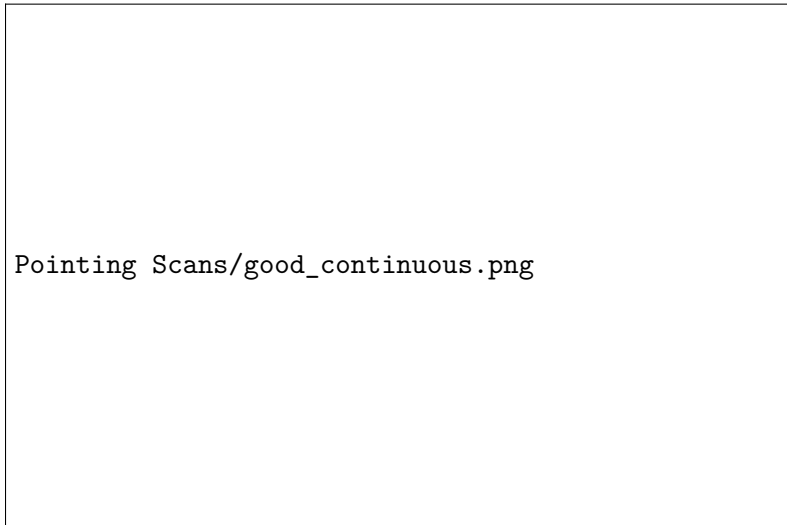


(b) Noisy line pointing with bad Gaussian fit.

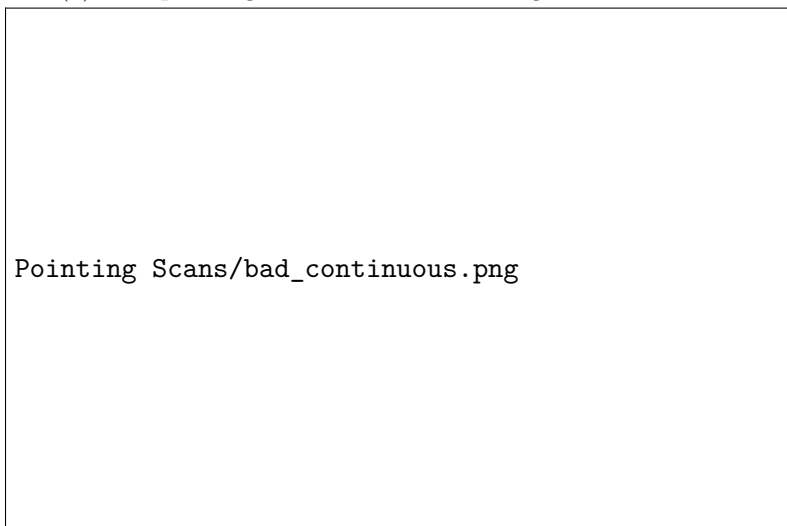
Figure 2.5: The two figures show line pointing scans. a) is good and clean, and b) is noisy and unreliable. A Gaussian is fit both for the azimuth and elevation pointing. The amplitude, full width at half maximum, offset, and the uncertainty of these measures are shown for both fits. The figures also show the correction that was used during the pointing (*ca* and *ie*), along with other metrics.

Continuum Scan Not all sources have emission lines, and for these sources a continuum scan is carried out instead. In this case, a source is continuously scanned in azimuth and elevation while recording the flux intensity. A Gaussian curve is

fitted to the recorded flux intensity to determine the offsets, amplitude, and full width at half maximum (FWHM). Figure 2.6 show examples of continuum scans and the corresponding Gaussian fits.



(a) Line pointing with little noise and a good Gaussian fit.



(b) Noisy line pointing with bad Gaussian fit.

Figure 2.6: The two panels show continuum pointing scans. a) is good and clean, and b) is noisy and unreliable. A Gaussian is fit both for the azimuth and elevation pointing. The amplitude, full width at half maximum, offset, and the uncertainty of these measures are shown for both of the fits.

Pointing scan timestamp In the main database, each pointing scan has a timestamp on the format YYYY-MM-DD HH:MM:SS, with second resolution. This timestamp does not reflect the the actual start of a pointing scan. Also, there is not information in the database itself that indicates whether the telescope is observing, but this information can be extracted from some dump files from the tiltmeter, which includes a flag indicating whether the telescope is idle, preparing to observe, or observing. Combining this flag with the timestamps, we can obtain the accurate start and end time of a pointing scan. However, these tiltmeter dump files are only available for some time periods.

- Plots with scan times distribution
- proportion of scans we have the accurate time for
- What we did for hte other scans
- mention some deviation

Instruments The telescope is equipped with various observing instruments that operates at a variety of frequencies. Table 2.3 shows the frequency band each of the instruments cover, along with the number of times they were used for a pointing scan throughout the year of 2022.

The broad range of frequencies at which astronomical phenomena emit electromagnetic radiation requires observation across a wide range of frequencies to study these phenomena comprehensively. A complete list of instruments and descriptions can be found at APEX’s website.

Table 2.3: The number of times each instrument was used for a pointing scan in 2022. There are 8847 scans in total.

Instrument	Frequency band [GHz]	# of scans
NFLASH230	200-270	3197
LASMA345	268-375	1861
NFLASH460	385-500	1394
SEPIA660	578-738	856
SEPIA345	272-376	818
SEPIA180	159-211	359
HOLO	-	225
ZEUS2	-	103
CHAMP690	-	34

2.6.4 Tiltmeter dump files

The tiltmeter dump files is a small part of the database, and are only used to analyze when pointing scans start and end. There are 280 of these files, and all have filenames on the format "Tiltmeter_YYYY-MM-DD.dump", which indicates which date the data is from. These files contrain 7 columns, which are datetime, azimuth, elevation, tilt1x, tilt1y, tilt1t and lastly the scan flag. For our purpose, only the datetime and scan flag are interesting. Table 2.4 show an extract of the datetime and scan flag columns from one of the tiltmeter dumps.

Table 2.4: Extract from a tiltmeter dump file.

Datetime	Scan flag
2022-11-13T02:23:37	IDLE
2022-11-13T02:23:38	IDLE
2022-11-13T02:23:39	PREPARING
2022-11-13T02:23:40	PREPARING
⋮	⋮
2022-11-13T02:23:52	PREPARING
2022-11-13T02:23:53	PREPARING
2022-11-13T02:23:55	OBSERVING
2022-11-13T02:23:56	OBSERVING
2022-11-13T02:23:57	OBSERVING

Chapter 3

Machine Learning Background

3.1 Supervised learning

Supervised learning is a subfield of machine learning that refers to training a model to predict a specific target value based on input data. In this context, the input data is commonly referred to as "features", and when paired with the corresponding target value for prediction, the training is supervised. There are two types of supervised learning, regression, and classification. In regression, we predict a continuous variable, while in classification predict a binary value, true or false. The model architecture of the model can be the same regardless of predicting a true/false or continuous value. The difference is in the loss function, which is used to evaluate the performance of the model during training, and for neural networks, the last layer activation function. In-depth explanations of this will come in the following sections.

3.2 Loss/Cost

In machine learning, the loss of a model refers to the discrepancy between the predicted and true values. It is calculated using a specific function designed to penalize incorrect predictions and serves as a measure of the model's performance. The ultimate goal of any machine learning model is to minimize the loss and thereby reduce the difference between predicted and desired outputs. To achieve this, the model is trained by calculating the gradient of the loss function with respect to different components in the model. These gradients are then used to determine how the model can be adjusted in the direction that minimizes the loss, through an iterative process. As a result, the model is optimized to make better predictions and achieve higher accuracy.

The most common loss function for regression is the mean squared error

$$\mathcal{L}(y, \tilde{y}) = \frac{1}{N} \sum_{i=1}^N (y - \tilde{y})^2 \quad (3.1)$$

3.2.1 Loss Functions

Loss functions are used to evaluate the performance of the machine learning model during training.

In this specific problem, when trying to predict the offset in both azimuth and elevation, there are two obvious ways to calculate the loss. One where the machine learning model considers the offset in azimuth and elevation separately, and one

where it considers both of them together. Let \tilde{y}_{Az} and \tilde{y}_{El} denote the prediction for the offset in azimuth and elevation respectively. y_{Az} and y_{El} are the true values. The first way to calculate is

$$\mathcal{L} = \frac{1}{N} \sum_i (y_{Az,i} - \tilde{y}_{Az,i})^2 + \frac{1}{N} \sum_i (y_{El,i} - \tilde{y}_{El,i})^2 \quad (3.2)$$

Using this loss function, the machine learning model will focus on minimizing both of the offsets evenly.

For the second loss function, we use the mean squared Cartesian distance

$$\mathcal{L} = \frac{1}{N} \sum_i \left[(y_{Az,i} - \tilde{y}_{Az,i})^2 + (y_{El,i} - \tilde{y}_{El,i})^2 \right], \quad (3.3)$$

which should have the effect of minimizing the offsets evenly.

3.3 Train/test set

Machine learning models can be highly complex and can fit all the data points in a dataset. While this can result in perfect predictions on the training data, it often leads to poor performance on new data, a phenomenon known as overfitting. To counteract this, the data is typically split into two parts - a training set and a test set (sometimes a third validation set is also used). The model is trained on the training set, and the error on the test set is used to evaluate the model's performance. By using a separate set of data for testing, we can get a better estimate of the model's performance on new data and avoid overfitting.

When the error on the training data is low, the model has low bias. However, if the model is too complex, it may also have high variance, meaning that it is overly sensitive to the training data and unable to generalize well to new data. A model with high variance may perform well on the training data, but its performance on new data may be poor. The key to building a good model is to balance bias and variance and to find the right level of complexity that will allow the model to generalize well. Proper selection of the train/test split ratio and the use of other techniques such as regularization can help achieve this balance and improve the model's performance.

3.4 Scaling

In machine learning, some models such as neural networks are highly sensitive to the scale of input data. The inputs to a model often contain different types of data with varying scales. Neural networks use weights to transform the input data, and each neuron in a fully connected network receives data from every input feature. If the input features have different scales, training the weights can be slow and unstable. Scaling the input data to have the same scale improves the speed and performance of the model. In contrast, tree-based models are not affected by the range scale of the data since they consist of tests and not mathematical operations.

The most common scaling method is to standardize the data to have zero mean and a standard deviation of one. This is achieved by subtracting the mean and dividing by the standard deviation. Mathematically, the standardization of a feature x is represented as:

$$x_{scaled} = \frac{x - \mu}{\sigma} \quad (3.4)$$

where μ is the mean of the feature values, and σ is the standard deviation of the feature values. The mean and standard deviation are computed using the following equations:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i \quad (3.5)$$

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2} \quad (3.6)$$

where n is the number of observations in the feature. In addition to standardization, other scaling methods such as min-max scaling and robust scaling are also used in specific cases. Overall, scaling is a crucial step in preprocessing data for machine learning, as it can significantly impact the performance of a model.

3.5 Decision Trees

Decision trees are tree-like models that make decisions based on conditions. As shown in Figure 3.1, each circle represents a node with various types, including decision nodes that split into two other nodes and leaf/terminal nodes that don't. The root node is the topmost decision node. Given an observation, there is a single path to a leaf node that represents the prediction made by the decision tree.

Trees are constructed greedily from the top, meaning that each split is made to minimize the loss function at the current step, without considering future splits. Single decision trees are not typically sufficient for complex problems. There are various methods to improve decision tree models, as demonstrated in Figure 3.2. The final step in the figure is XGBoost (Extreme Gradient Boost), a highly efficient and high-performing machine learning algorithm. In this section, we'll briefly cover the methods used to optimize decision trees for prediction. [6]

Bagging Bagging, also known as Bootstrap Aggregation, is a method for training an ensemble of models that contribute to the final prediction. Each model is trained using bootstrapped data (resampled from the original dataset with replacement), resulting in diverse decision trees. The final prediction is the average of all ensemble models.

Random Forest A method based on bagging, where each tree in the ensemble is made using only a randomly chosen subset of features. This often leads to better generalization and reduced overfitting.

Boosting An ensemble is created in boosting as well, but the trees are not made independently. They are trained one by one considering the previous trees. A sample weight is assigned to each sample used to train a tree based on the current ensemble's accuracy. Samples with large prediction errors are assigned larger weights and those with accurate predictions are assigned lower weights. The final prediction is a weighted sum of all ensemble predictions, with weights based on each individual tree's accuracy.

Gradient Boosting Like in regular boosting, an ensemble of trees is created iteratively by considering the errors made by previous trees. The process starts with a constant model that predicts the mean of all samples. The gradient of the loss function with respect to each sample is calculated and a tree is made to predict these gradients. The new prediction is the constant plus a small step in the direction of the predicted gradients. Repeated iteration with small steps in the gradient direction helps reduce both bias and variance.



Figure 3.1: Decision tree with 3 decision nodes and 5 leaf nodes.

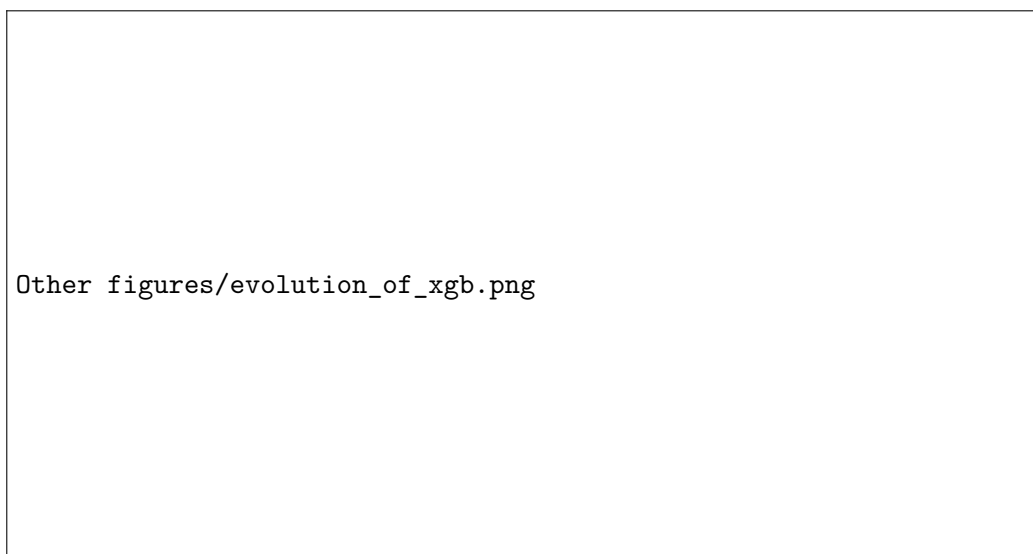


Figure 3.2: Evolution of XGBoost. [7]

3.6 Neural Networks

A Neural Network (NN) is an Artificial Intelligence (AI) model composed of interconnected neurons, inspired by biological neural networks found in animal brains. NNs are arranged in layers, as shown in Figure 3.3, and consist of an input layer, one or more hidden layers, and an output layer. The size of the hidden layer(s) varies depending on the nature of the problem. NNs process input to produce an output, which ideally is close to what you aim to predict.

Each connection in an NN has a trainable weight w_{jk}^l , representing the weight from the k^{th} neuron in layer $(l-1)$ to the j^{th} neuron in layer l . Each neuron also has its own bias b_j , added to its output to prevent the input to its activation function σ from being zero. The activation function σ , applied to the neuron's output, is the final transformation before passing data to the next layer. This nonlinear function is crucial in allowing NNs to learn nonlinear relationships in data [1].

The following is the mathematical explanation of how a neuron processes the outputs from the previous layer.

$$a_j^l = \sigma \left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l \right) = \sigma(z_j^l) \quad (3.7)$$

The quantity

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \quad (3.8)$$

will be useful in explaining how to optimize a neural network, and can be considered the weighted input for neuron j in layer l .

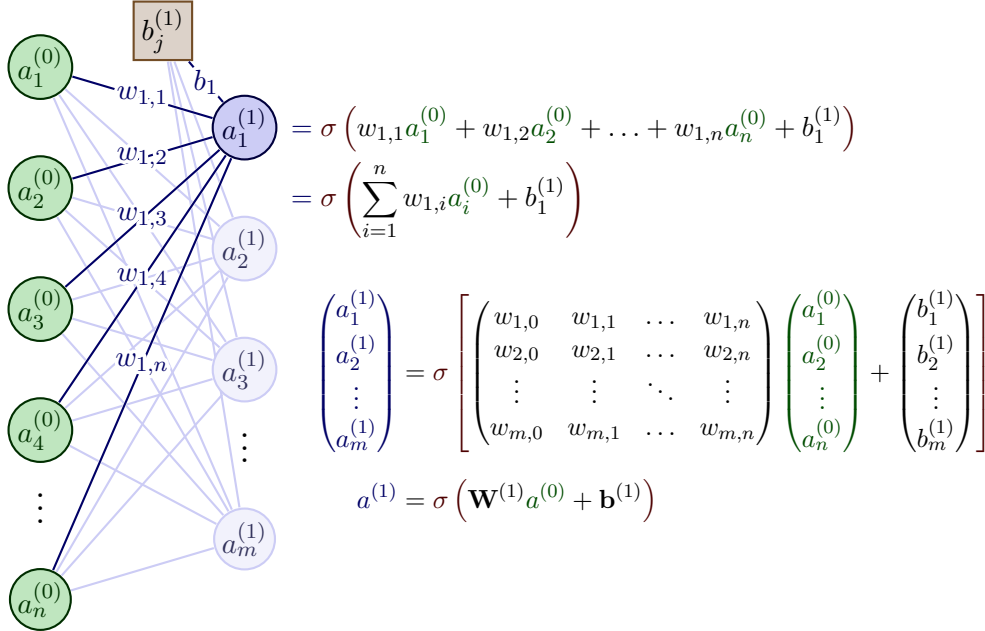


Figure 3.3: This is an illustration of how information is passed through and processed in a neural network. Generated using TikZ [11]

3.6.1 Backpropagation

Backpropagation[9] is a fundamental algorithm in training artificial neural networks. It calculates the gradient of the loss function with respect to all the weights and biases

in the network, allowing for the updating of these parameters in a way that reduces the loss. The algorithm is based on four key equations, which will be described in this section.

We define the error in the j^{th} neuron in the l^{th} layer by

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial a_j^l} \sigma'(z_j^l) \quad (3.9)$$

This can also be considered the partial derivative of the cost function with respect to the bias in neuron j in layer l , as

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l} \frac{\partial b_j^l}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l}, \quad (3.10)$$

where we have used the relation $\partial b_j^l / \partial z_j^l = 1$ from rearranging equation (3.8). The next equation relates the error in a neuron with the errors in the neurons in the subsequent layer.

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \left(\sum_k \delta_k^{l+1} w_{kj}^{l+1} \right) \sigma'(z_j^l) \quad (3.11)$$

Note that the indices on the weight w are now swapped. We may think of this equation as an error propagating backward by multiplying the error in layer $l + 1$ with the transpose of the weight connecting layer l with $l + 1$. The final equation is derived from the partial derivative of the cost function with respect to the weight w_{jk}^l

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1} \quad (3.12)$$

Equation (3.9) lets us calculate the error in the last layer, and using equation (3.11) we can propagate this error backward through the network, calculating the error for all the neurons. We then use equations (3.10) and (3.12) to calculate the gradient of the cost function with respect to the weights and biases.

3.6.2 Activation functions

Activation functions play a crucial role in the training of a neural network by allowing it to learn non-linear relationships between inputs and outputs. Different activation functions have varying properties, and some of the most common ones will be discussed in this section. Properties like non-linearity, differentiability, monotonicity, smoothness, and zero-centering are considered important for activation functions. Non-linearity enables the model to capture complex relationships, differentiability is necessary for calculating the derivative of the loss function with respect to the trainable weights, monotonicity helps ensure stability in activation outputs, smoothness stabilizes gradients during training, and zero-centering balances the activation distribution within the model.

Activation functions play a crucial role in the training of a neural network by allowing it to learn non-linear relationships between inputs and outputs. Different activation functions have varying properties, and there are some properties considered important.

- Non-linearity enables the model to capture complex relationships

- Differentiability is necessary for calculating the derivative of the loss function with respect to the trainable weights
- Monotonicity helps ensure stability in activation outputs, smoothness stabilizes gradients during training
- Smoothness: A smooth activation function helps with making the gradients and training stable.
- Zero-centering balances the activation distribution within the model.

Tanh Tanh, the hyperbolic tangent function is given by

$$\text{Tanh}(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}} \quad (3.13)$$

ReLU The Rectified Linear Unit (ReLU) activation function pushes all negative values to zero while leaving positive values unchanged. This introduces non-linearity while solving the vanishing gradients problem by having a gradient of either 0 or 1 for negative and positive values, respectively.

$$R(x) = \begin{cases} x & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \quad (3.14)$$

3.7 Model Explainability

In the context of machine learning, SHAP [5] and SAGE [3] apply the same idea to determine the contribution of each feature to a prediction. SHAP provides a local explanation by computing the contribution of each feature to the prediction of a single data point. On the other hand, SAGE provides a global explanation by computing the contribution of each feature to the overall prediction performance of the model. These methods allow us to understand the relationship between the features and the prediction, which is particularly useful when the model is too complex to interpret. Additionally, they provide a way to validate the model's fairness and bias. By understanding which features are contributing the most to a prediction, one can determine if the model is fair or biased, and if the prediction is trustworthy.

Both SHAP and SAGE methods are based on Shapley values [10], a concept in game theory introduced by Lloyd Shapley in 1951. Shapley values determine the contribution of each player to a group's surplus or overall value. The explanation below of Shapley values, SHAP, and SAGE is inspired by a blog post by Ian Covert [2].

The Shapley value for a player i in a cooperative game with d players is

$$\phi_i(w) = \frac{1}{d} \sum_{S \subseteq D \setminus \{i\}} \binom{d-1}{|S|}^{-1} [w(S \cup \{i\}) - w(S)] \quad (3.15)$$

where D is the set of all players, S is a coalition of players, $w(S)$ is the value of the coalition S , and $|S|$ is the number of players in the coalition. This formula satisfies four important conditions:

- Efficiency: The sum of all Shapley values should be equal to the total value of the group.

- Symmetry: If two players i and j have the same impact on all coalitions with $w(S \cup \{i\}) = w(S \cup \{j\})$ for all S , they should have the same Shapley value $\phi_i(w) = \phi_j(w)$.
- Dummy: A player i that makes no contribution to the group with $w(S \cup \{i\}) = w(S)$, should receive a value of zero, or $\phi_i(w) = 0$.
- Linearity: The value of a player should be proportional to their contribution to the group. If player i contributes twice as much as player j to the group's overall worth, then player i should have twice the Shapley value.

3.7.1 SHAP

Shapley values explain how each feature (x^1, \dots, x^d) in a model f contributes to the deviation from the mean prediction $\mathbb{E}[f(x)]$ of the dataset, for a single prediction. It assigns a value ϕ_1, \dots, ϕ_d to each feature, that quantifies the feature's influence on the prediction $f(x)$. SHAP (Shapley Additive Explanations) is a way of computing approximate Shapley values for machine learning models.

We define a cooperative game $v_{f,x}$ to represent a prediction given the features x^S , as

$$v_{f,x}(S) = \mathbb{E} \left[f(X) | X^S = x^S \right], \quad (3.16)$$

where x^S are known, and the remaining features are treated as random variable $X^{\bar{S}}$ (where $\bar{S} = D \setminus S$). This is the mean prediction $f(X)$ when the unknown values follow the conditional distribution $X^{\bar{S}} | X^S = x^S$.

Using a subset of features from the prediction while sampling the rest from the dataset, reduces the chance of improbable samples. Given this convention for making predictions, we can apply the Shapley value to define each feature's contribution to the prediction $f(X)$ using Shapley values $\phi_i(v_{f,x})$. A Shapley value of $\phi_i(v_{f,x}) > 0$ indicates that feature i contributes to an increase in prediction $f(X)$. A negative Shapley value $\phi_i(v_{f,x}) < 0$ indicates the opposite, that the feature contributes to a decrease in $f(X)$. Uninformative features will have small values $\phi_i(v_{f,x}) \approx 0$.

3.7.2 SAGE

SAGE (Shapley Additive Global Importance) explains how every feature contributes to the performance of the model overall, and it relates to SHAP in a simple way. For a given feature, the global feature importance is the average SHAP value (for that feature) across all samples in the dataset. This is, however, not how it is calculated in practice. A paper by Ian Covert et al. [4] on global feature importance proposes an algorithm that aims directly at a global feature explanation, unlike the SHAP values, which makes it faster. This is the algorithm used for approximating the SAGE values for the features in the thesis.

Chapter 4

Method

In this section, I explain the data analysis, cleaning, transformation, feature engineering, and models.

4.1 Cleaning pointing scan data

4.1.1 Cleaning rules

The pointing scan data is noisy, and contains a lot of outliers. In order to make the data more suitable for machine learning, we need to clean the data. Below is a list of some criteria we use to filter out scans.

- Scans using the HOLO transmitter. These scans are aiming at a radio tower, and is therefore not realistic data to train the ML model with.
- Scans using ZEUS2. These are highly experimental pointing scans and unreliable.
- Scans using CHAMP690, as there are very few scans using this instrument.
- Scans in January and February. There are not many scans in this time period, and the few we have involve a lot of testing.
- Scans that are tracking tests.

Find the proportion of scans that are removed from the above list. After this filtering, there are x out of 8862 scans left.

4.1.2 Pointing scan classifier

Method

In addition to the cleaning rules, we have to remove the outright bad pointing scans (like 2.6b 2.5b, 2.5b). This is done by training a classifier predict whether a scan is good or bad. We used an XGBoost classifier with 13 features as inputs, all of which are present in the pointing scan figures (2.5 and 2.6). The first 12 features are the amplitudes, FWHMs and pointing offsets, along with the uncertainties of these values. The last feature is the beamsize of the telescope for the given observing frequency.

We had to label a training set by manually looking at pointing scans. The size of the training set was 369 with 270 good and 99 bad scans. We did a hyperparameter search for the max depth between 1 and 5, number of estimators between 1 and 80,

and used the *scale_pos_weight* to consider the unbalanced classes. The value for this parameter is the ratio of negative to positive classes. 80% of the data was used for training, and the rest for testing, which corresponds to 295 and 74 samples for training and testing respectively.

- What hyperparameters were tuned, and what are the values.
- Show why it is hard to clean by only noise
- labeling training set manually, size and proportion of pos neg
- can choose how strict we want to be on the cleanliness
- precision recall curve or mAP curve?

Results

The XGBoost classifier performed well a 97% overall accuracy on the test set. Figure 4.1 shows the precision-recall curve on the left and the average precision curve on the right. From the precision-recall curve it is clear that we can achieve a close to 100% precision while still having a high recall. This is ideal, as we want the training data for the pointing model to be as clean as possible. We can select a large threshold, such that very few bad scans end up in the training data. The average precision curve shows an optimal threshold for maximizing the precision, which is about 80%.

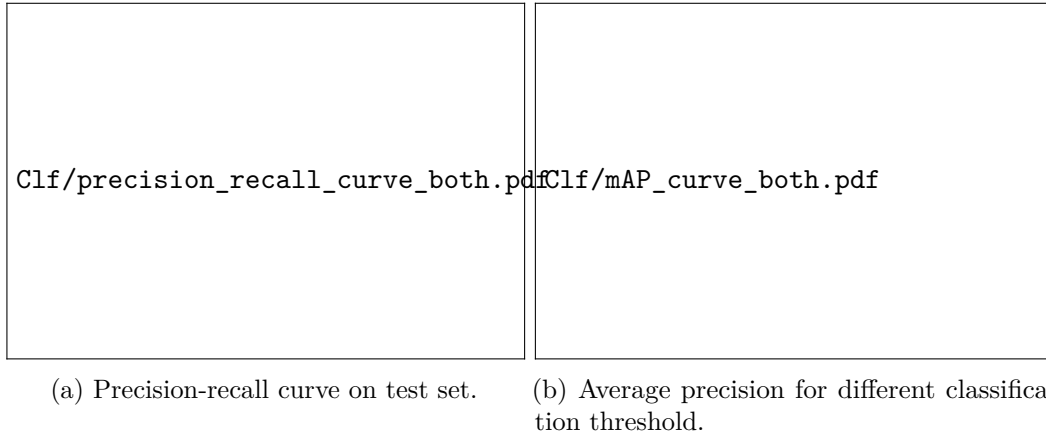


Figure 4.1: Precision-recall and average precision curve for the XGBoost classifier when classifying good and bad pointing scans in the test set.

4.2 Scan duration analysis

As mentioned in the database section, the timestamps of the scans is not the accurate start time of a scan. The tiltmeter dump files with the flag indicating whether the telescope is idle, preparing to observe, or observing, is the only data we have on when the telescope is actually performing a pointing scan. Therefore, we need to combine the timestamp of the pointing scan with the flag in the dump files to analyse the duration of scans.

4.2.1 Method

First, we convert the different scan flags to numbers. *IDLE* and *PREPARING* is the set 0, and *OBSERVING* is set to 1. Then we can subtract the previous rows from all rows, which will result in a value of 1 when the scan starts, and -1 when it ends. Table 4.1 shows an example of the resulting table.

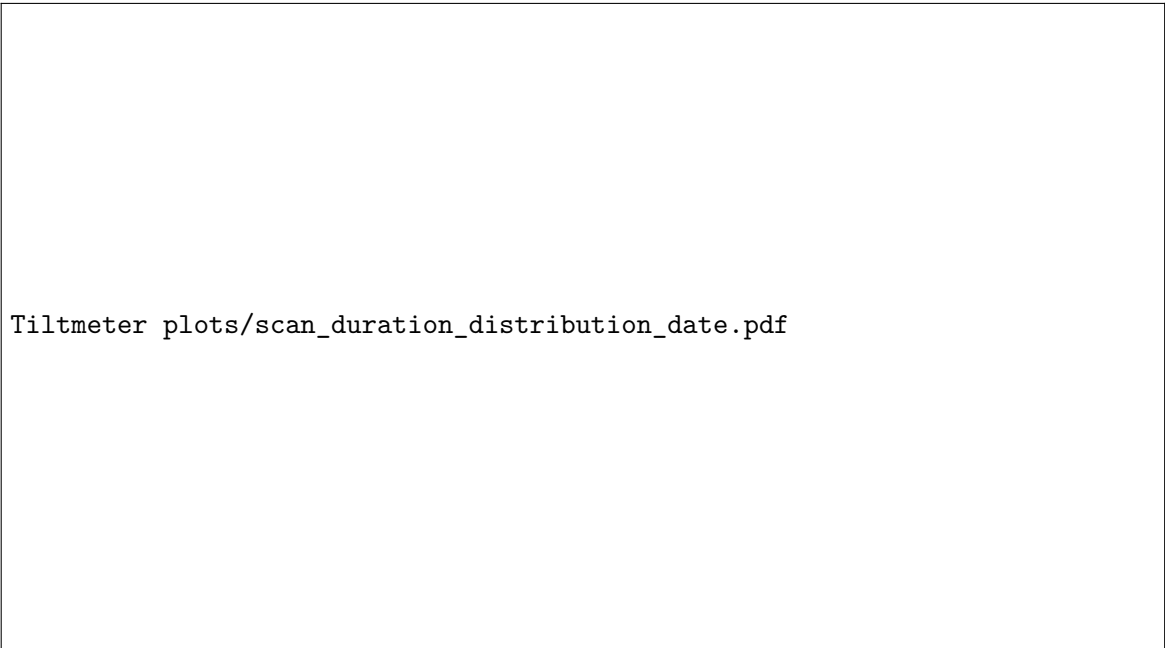
Time	Flag	Flag Integer	Δ
11:21:21	IDLE	0	0
11:21:22	PREPARING	0	0
11:21:23	OBSERVING	1	1
11:21:24	OBSERVING	1	0
11:21:25	OBSERVING	1	0
11:21:26	IDLE	0	-1

Table 4.1: This table shows the tiltmeter dump file containing the telescope state flag, and how we find the start ($\Delta = 1$) and end ($\Delta = -1$) of a scan.

By analyzing these tables for all the scans we had tiltmeter dumps for, we see that the first *OBSERVING* flag present after a scan is about 53 **Add accurate mean and standard deviation** seconds after the scan timestamp for the vast majority of the scans. This is shown in the right plot in figure 4.3, which strongly indicates that this is the starting point of a pointing scan. In the same plot, we also see that this is fairly constant for the different instruments. The right plot of figure 4.2 show the time difference in seconds between the first observing flag after a scan timestamp throughout the year. From the plot, we may conclude that this stays constant over time.

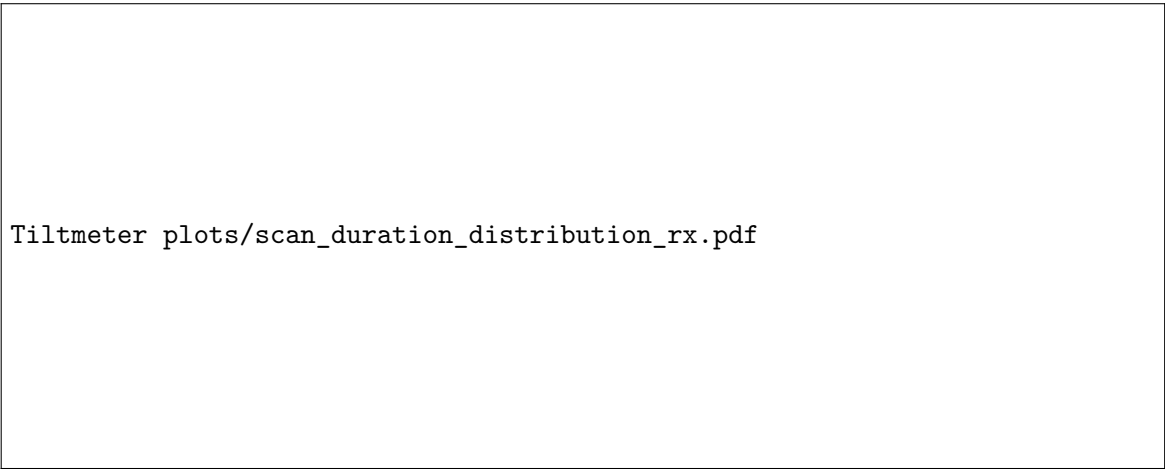
Now that we have found the start of a pointing scan, we look at the durations of the scans.

The left plots in figure 4.2 and 4.3 show the duration of the pointing scans for different instruments. From this figure, it is clear that the duration of a pointing scan varies a lot. This is problematic because of the fact that we only have these tiltmeter dump files for $2862/8381 \approx 34\%$ of the pointing scans.



Tiltmeter plots/scan_duration_distribution_date.pdf

Figure 4.2: Scatter plot of the duration of scans, and the time difference between the timestamp of a scan and the actual start of it.



Tiltmeter plots/scan_duration_distribution_rx.pdf

Figure 4.3: Box plot of the duration of scans, and the time difference between the timestamp of a scan and the actual start of it.

4.2.2 Algorithm

Following is the algorithm used to obtain the start and end of pointing scans using the scan timestamp and the observing flag from tiltmeter dumps.

Algorithm 1 Find start and end of pointing scan

Input:

- Pointing scan timestamps $D = \{D_1, \dots, D_n\}$
- Timestamps $T = \{T_1, \dots, T_m\}$ and scan flag $F = \{F_1, \dots, F_m\}$

Output: Start and end of pointing scans $S = \{S_i, \dots, S_n\}$ and $E = \{E_i, \dots, E_n\}$

```
for  $i = 1, \dots, m$  do
  if  $F_i = \text{OBSERVING}$  then
     $F_i = 1$ 
  else
     $F_i = 0$ 
  end if
end for

for  $i = 1, \dots, n$  do
   $\hat{T} = \{T_j, \text{ if } T_j > D_i\}_j^m$ 
   $\hat{F} = \{F_j, \text{ if } T_j > D_i\}_j^m$ 
  for all  $t_i, f_i$  in  $\hat{T}, \hat{F}$  do
     $\Delta = f_i - f_{i-1}$ 
    if  $\Delta = 1$  then
       $S_i = t_i$ 
    end if
    if  $\Delta = -1$  then
       $E_i = t_i$ 
      Continue
    end if
  end for
end for
```

4.3 Feature Engineering

4.3.1 Different calculations

There are two main types of features engineered for this project. Features that represent the system during a pointing scan, and features that represent the changes since the last correction. The idea behind this is simple. The correction used during a pointing scan represents the ideal correction for the system during the previous pointing scan. As there are a lot of factors and complex relationships, and we don't have large amounts of training data, it might be easier for the model to learn how these changes affect the pointing rather than learning all the relationships.

Median values The median value of variables during a pointing scan is the most used feature. Table [ref table with list of variables with median value](#) shows the list of variables for which we use the median values during a pointing scan.

Sum of all change To capture systematic error in pointing due to the telescope moving back and forth in azimuth and elevation, we sum over the positive and negative changes in these variables.

Given the time of the last pointing correction t_1 and the start of a pointing scan t_2 , the sum over the positive changes in a variable x_i is given by

$$\sum_{i=t_1+1}^{t_2} \max(0, x_i - x_{i-1}) \quad (4.1)$$

Similarly, the sum of negative changes in a variable is

$$\sum_{i=t_1+1}^{t_2} \min(0, x_i - x_{i-1}) \quad (4.2)$$

These features are made for azimuth and elevation.

Change since last correction This feature is self-explanatory and is just the change in a variable since the pointing was corrected.

$$\Delta x = x_{t_2} - x_{t_1} \quad (4.3)$$

In order to make this feature more robust against noisy data, we instead look at the change in the median for a time interval around the last correction t_1 and the start of a pointing scan t_2

$$\Delta x = \text{median}(x_{t_2}, x_{t_2-1}, \dots, x_{t_2-p}) - \text{median}(x_{t_1}, x_{t_1+1}, \dots, x_{t_1+p}), \quad (4.4)$$

where p is the number of data points needed to cover a period of P minutes, given by $p = P \cdot \text{frequency}$. The unit of frequency is data points per minute, found in table [fix data freq ref](#).

Max change in time interval In case the speed of the temperature change affects the deformation of the telescope's structure, we find the maximum temperature change in a given time interval since the last pointing correction.

$$\max(x_{t_1+p} - x_{t_1}, x_{t_1+p} - x_{t_1}, \dots, x_{t_2} - x_{t_2-p}), \quad (4.5)$$

Position of the sun Observers at the telescope report that the sun is affecting the pointing. It is most drastically affected when the sun sets or rises, likely due to rapid change in temperature which leads to deformation in the telescope structure. It is also thought that the position of the sun could play a role, and perhaps be modelled too. For instance, if the sun is shining on the left side of the telescope, it will affect the pointing differently than if it was on the right side. Obtaining the position of the sun for the location of the telescope is done using the python module PyEphem [8].

Using the azimuth angle of the sun and the telescope, we can calculate the position of the sun with respect to the pointing with

$$\Delta Az_{\odot} = Az_t - Az_{\odot} \quad (4.6)$$

This will result in values outside the $[-180^\circ, 180^\circ]$. An example of this is if $Az_{\odot} = 179^\circ$ and $Az_t = -179^\circ$. The calculation in equation (4.6) yield $-179^\circ - 179^\circ = -358^\circ$, which corresponds to the sun being 358° to the right of the telescope, while it ideally should be 2° to the left. Therefore, we adjust the values accordingly

$$\Delta Az_{\odot} = Az_{\odot} + 360^\circ, \text{ for } \Delta Az_{\odot} < -180^\circ \quad (4.7)$$

$$\Delta Az_{\odot} = Az_{\odot} - 360^\circ, \text{ for } \Delta Az_{\odot} > 180^\circ \quad (4.8)$$

Here, the interval of the difference in azimuth is fixed to the interval $(-180^\circ, 180^\circ)$, where 0° means the telescope is pointing towards the sun in the azimuth direction. $\Delta Az_\odot = 90^\circ$ corresponds to the sun being direct to the left of the pointing direction.

Another measure tested is the total angle between the pointing and the position of the sun. This is calculated using the following formula

$$\theta = \cos Az_t \cdot \cos El_t \cdot \cos Az_\odot \cdot \cos El_\odot + \sin Az_t \cdot \cos El_t \cdot \sin Az_\odot \cdot \cos El_\odot + \sin El_t \cdot \sin El_\odot \quad (4.9)$$

4.3.2 List of features

[add a list of all features for different calculations here](#)

4.4 Transformation of pointing offsets and corrections

Table 4.2 shows some observed pointing offsets and the corrections applied during the pointing scan. The pointing corrections *ca* and *ie* are sometimes updated according to equation (2.17) and (2.18), as we see in the first two rows. Sometimes the corrections are not updated, like in the consecutive row. Other times, the corrections are updated, but not according to the equation (2.17) and (2.18). This introduces a couple of challenges for the training of a model.

- *ca* and *ie* should represent the optimal correction using all the information we have about the current state of a system. If we don't update the corrections, there is some information about the system (the previously observed pointing offset) the model is not receiving.
- Some of the features are constructed by looking at the change in variables since the last correction. If the corrections are not updated, this interval is longer and the resulting features could be more prone to uncertainties and noise. A problem with the integration also occurs if the corrections are not updated according to the equations (2.17) and (2.18). Then we don't know at what time those corrections represent the system, resulting in inaccurate features.

The following is a possible solution to these problems.

1. Transform the offsets and corrections such that they represent the system at the most recent pointing scan.
2. Use these transformed offsets as new training labels, and transformed corrections as training inputs.

The way I do this is by assuming the corrections *ca* and *ie* are updated after every pointing scan. This will change the observed pointing offsets, which again will affect the corrections during the next pointing scan. The effects of this followed throughout the whole dataset. The following formulas

$$\tilde{ca}_i = \tilde{ca}_{i-1} + \tilde{\delta}_{az,i-1} \quad (4.10)$$

$$\tilde{ie}_i = \tilde{ie}_{i-1} - \tilde{\delta}_{el,i-1} \quad (4.11)$$

$$\tilde{\delta}_{az,i} = \delta_{az,i} + ca_i - \tilde{ca}_i \quad (4.12)$$

$$\tilde{\delta}_{el,i} = \delta_{el,i} - ie_i + \tilde{ie}_i \quad (4.13)$$

Where the " \sim " denotes a transformed variable. Using these transformations, the corrections used for training and the resulting offset are alike the ones that would be observed if the corrections were made according to the equations (2.17) and (2.18) after every pointing scan.

Table 4.2: Example from the dataset of the observed pointing offsets and the corrections applied during the pointing scan.

i	δ_{az}	δ_{el}	ca	ie
1	1.2	0.1	2.1	1.7
2	0.0	0.5	3.3	1.6
3	-1.1	0.0	3.3	1.6
4	0.6	0.7	2.2	1.6
5	0.9	1.4	2.2	1.6
6	1.0	1.1	2.2	1.6
7	-0.9	1.2	3.1	0.5
8	0.5	1.5	2.2	-0.7
9	-0.3	0.4	2.2	-0.7

Table 4.3: Table of transformed pointing offsets and corrections according to equations (4.10), (4.11), (4.12), and (4.13).

i	$\tilde{\delta}_{az}$	$\tilde{\delta}_{el}$	\tilde{ca}	\tilde{ie}
0	1.2	0.1	2.1	1.7
1	0.0	0.5	3.3	1.6
2	-1.1	-0.5	3.3	1.1
3	0.7	0.7	2.2	1.6
4	0.2	0.7	2.8	0.9
5	0.1	-0.3	3.1	0.2
6	-1.0	1.3	3.2	0.5
7	0.5	1.4	2.2	-0.7
8	-0.8	-1.1	2.7	-2.2

4.5 Experiments overview

In this section, I will provide an overview of two machine learning experiments that are pertinent to the two research questions. The first experiment aims to investigate the effectiveness of neural networks in developing a pointing model that could potentially replace the current linear model, which is created through linear regression. The purpose of this experiment is to explore the feasibility of a more sophisticated model in terms of pointing accuracy. The second experiment aims to examine the effectiveness of an XGBoost model in predicting pointing scan offsets to enhance the pointing accuracy. The primary objective of this experiment is to assess whether the proposed model can outperform the current model in terms of pointing accuracy.

4.6 Experiment 1: Pointing Model using Neural Networks

This experiment uses the raw dataset containing input coordinates, Az_{input} and El_{input} respectively, and corresponding true observed values Az_{observed} and El_{observed} .

The goal is to find a model f such that

$$f(X) \approx (\delta_{Az}, \delta_{El}) = (Az_{\text{observed}} - Az_{\text{input}}, El_{\text{observed}} - El_{\text{input}}) \quad (4.14)$$

4.6.1 Feature Selection

Selecting the right features plays an important role in improving the accuracy of the pointing model. This model uses two types of features: geometrical and harmonic terms that are already part of the current linear base model, and new features that are extracted from the telescope’s database. We identified relevant features by calculating Pearson’s and Spearman’s rank correlation for all features in relation to the offsets. We analyzed the correlation of harmonic terms using sine and cosine functions of azimuth and elevation up to the fifth order, as well as the geometrical terms. Then, we chose the terms that had the strongest correlation for the model and used them in all models. For the features extracted from the database, we made a list of the ones that showed a correlation equal to or greater than 0.1. During model training, we randomly selected a subset of 2 to 19 features from this list and used them to train the model.

4.6.2 Model Architecture

This experiment utilized four different model architectures. The first architecture involved feeding all of the input data into one, two, or three hidden layers. The other three architectures incorporated machine learning techniques by separating the geometrical and harmonic terms of the input data from the other features and processing them using distinct architectures. These approaches were intended to keep the current model’s simplicity and performance, while still incorporating new features.

The following are the four different architectures:

1. **Neural Network with Separated Features 1:** This architecture separates the input features into two groups: geometric and harmonic features, and the rest of the features. The geometric and harmonic features are connected directly to the output layer, while the remaining features are passed through layers with non-linear activation functions.
2. **Neural Network with Separated Features 2:** This architecture is similar to the previous architecture, but the geometric and harmonic features are passed through an additional layer of non-linear activation function before connecting them to the output layer.
3. **Neural Network with Separated Features 3:** This architecture combines the previous two architectures by passing the regular features through a few hidden layers with non-linear activation functions before concatenating them with the geometric and harmonic features. The combined features are then passed through a final layer before connecting to the output layer.

4. **Regular Neural Network:** All features are passed through the same layers, all with a nonlinear activation function.

These are visualized in figure 4.4.

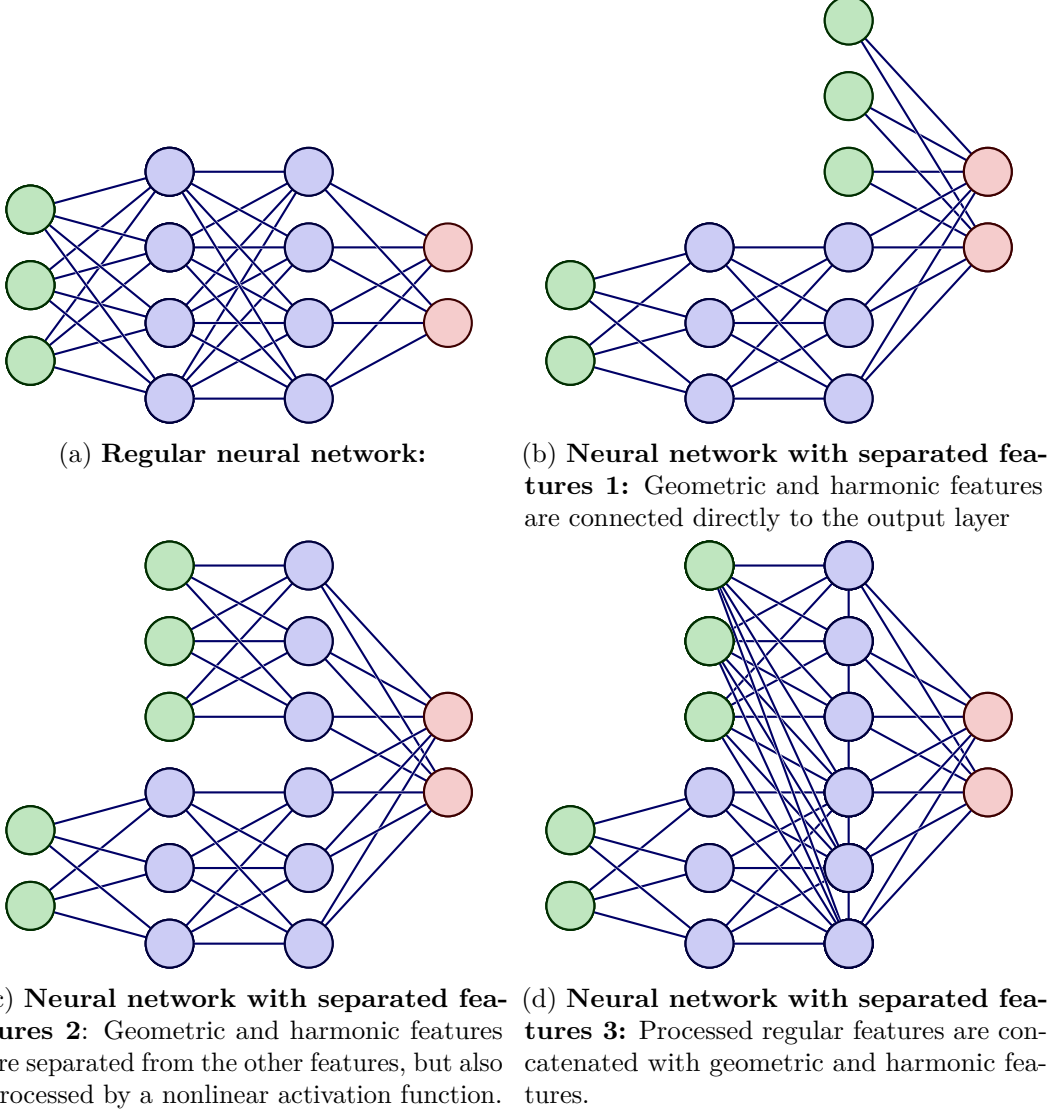


Figure 4.4: Neural network architecture for the base pointing model trained on raw data.

The hyperparameters for the neural networks were sampled from different distributions, as presented in Table 4.4. While some parameters were consistent across all models, such as the use of the Adam optimization algorithm, and the mean squared error loss function.

Table 4.4: This table presents a list of parameters that are sampled during hyperparameter tuning for the base pointing model. The table includes names, distributions that are sampled from, and corresponding ranges.

Name	Distribution Type	Range
hidden layers	uniform integer	[1,3]
hidden layer size	uniform integer	[20, 120]
learning rate	uniform	[0.001, 0.02]
batch size	uniform integer	[32, 512]
activation	categorical	[gelu, tanh]

4.6.3 Model Evaluation

4.7 Experiment 2: Pointing Correction Model

In this experiment, we aim to improve the pointing accuracy by training XGBoost models to predict offsets obtained by pointing scans. We use four different datasets, all of which are preprocessed using our cleaning criteria and the XGBoost classifier trained to remove bad pointing scans. These datasets use

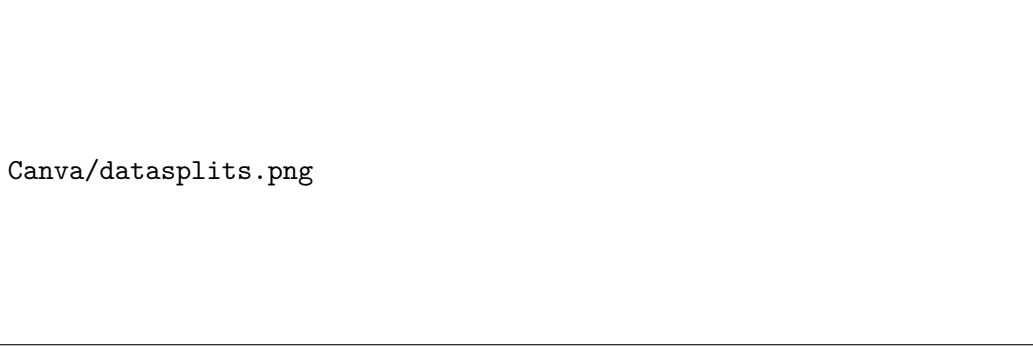
- all instruments
- NFLASH230 only
- all instruments transformed to simulate optimal pointing corrections
- NFLASH230 only transformed to simulate optimal pointing corrections

By training our models on these datasets, we hope to reduce the pointing offset and improve the accuracy of the pointing.

In addition, we varied the way we split the datasets for training and testing. We considered two cases:

- **Case 1:** The dataset is sorted by date and split into six equal-sized folds. We consider each of the folds one by one. For each of these folds, we use the last 1/6th of the data set test, and the remaining 5/6th as training and validation.
- **Case 2:** The dataset is sorted by date and split into six equal-sized folds. We used 5/6 of the data for training and validation and the remaining for testing. We repeated this process six times, using each fold for testing once.

Figure 4.5 illustrates the two cases. In both cases, we trained and validated the model on 5/6 of the data and tested on the last 1/6. The difference is the amount of data used for training, which can indicate whether models trained on shorter or longer periods perform better.



Canva/datasplits.png

Figure 4.5: Two cross-validation cases are shown, where the orange region represents the train and validation set, the red region represents the test set, and the blue region is unused for evaluation. In **Case 1**, the dataset is split into six equal-sized folds sorted by date. For the selected fold, the last part (colored red) is used for testing, and the remaining part (colored orange) is used for training and validation. This process is repeated six times, once for each fold. In **Case 2**, the dataset is again split into six equal-sized folds sorted by date. However, this time, one whole fold is used for testing, and the remaining five folds are used for training and validation. This process is repeated six times, with each fold used exactly once for testing.

4.7.1 Feature Selection

We trained models using a range of features, specifically $k = [2, 5, 10, 20, 30, 40, 50]$ features. For each model, we selected the k features that had the greatest mutual information with the target value. This approach helps us identify the most important features and can improve the model’s performance by reducing overfitting and noise. By selecting different numbers of features, we can explore the trade-off between model complexity and performance. [write and ref to mutual info in theory section.](#)

4.7.2 Model Architecture

For each model, we performed a Bayesian hyperparameter search using the parameter space shown in Table 4.5. The search space includes eight hyperparameters that affect the model’s complexity, such as the maximum depth of the trees, the regularization strength, and the learning rate. We used a uniform or log-uniform distribution to sample each hyperparameter within a specific range. In total, we evaluated 200 different combinations of hyperparameters (for each dataset, cross-validation case, target variable, and the number of features selected) to find the optimal values for each model. The models were validated using the MSE, and the model with the best performance on the validation set was picked.

Table 4.5: This table presents a list of parameters that are sampled during hyperparameter tuning for the pointing correction model. The table includes names, distributions that are sampled from, and corresponding ranges.

Parameter	Sample Distribution	Range
max depth	Uniform	[1, 5]
reg lambda	Uniform	[0, 1]
colsample bytree	Uniform	[0.5, 1]
n estimators	Uniform	[20, 500]
learning rate	Log-Uniform	$[10^{-5}, 1]$
subsample	Uniform	[0.5, 1]
gamma	Log-Uniform	$[10^{-5}, 1]$
min child weight	Uniform	[1, 10]

4.7.3 Model Evaluation

To evaluate the performance of the models, we calculated the RMS on each test fold and compared it to the current RMS of the telescope on the same data. We then computed the ratio of the model’s RMS to the current RMS for each fold, denoted as $r_{RMS,i}$. If the ratio is less than 1, it indicates that the XGBoost model provides an improvement over the current performance of the telescope on that fold.

To obtain an overall measure of the model’s performance compared to the current performance of the telescope, we averaged the ratios $r_{RMS,i}$ over all six test folds

$$\bar{r}_{RMS} = \sum_{i=1}^6 \frac{RMS_{model,i}}{RMS_{current,i}}. \quad (4.15)$$

This gives us an average ratio denoted as \bar{r}_{RMS} , which is a measure of the improvement in performance provided by the XGBoost model. If $\bar{r}_{RMS} < 1$, it indicates that the XGBoost model outperforms the current pointing correction method on average across all test folds. By comparing the average ratio \bar{r}_{RMS} for the two different cross-validation cases and the selected number of features, we can identify which models provide the best performance.

Bibliography

- [1] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.
- [2] Ian Covert. Understanding shap and sage. <https://iancovert.com/blog/understanding-shap-sage/>, 2020.
- [3] Ian Covert, Scott Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures, 2020.
- [4] Ian Covert, Scott Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures, 2020.
- [5] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.
- [6] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, may 2019.
- [7] Vishal Morde. Xgboost algorithm: Long she may rein. Downloaded January 2023.
- [8] Brandon Rhodes. Ephem, 2021. <https://pypi.org/project/ephem/>.
- [9] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [10] Lloyd Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2:307–317, 1953.
- [11] Till Tantau. The PGF/TikZ manual, 2021.
- [12] Tomruen. Azimuth-altitude schematic, 2011. Accessed: March 13, 2023.
- [13] Tpoint Software. *TPOINT*, 2009. Version 13.5.