Machine Learning for Anti-Poaching: Decision Tree Applications on the Savannah

Jakob Olof Popov Wirén

Kasper Nordenram



LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6269
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

© 2025 Jakob Olof Popov Wirén & Kasper Nordenram. All rights reserved.
Printed in Sweden by Tryckeriet i E-huset
Lund 2025

Abstract

The use of technology, specifically machine learning, to address complex issues in developing regions has become increasingly prominent. In the context of wildlife conservation, combating poaching has emerged as a significant application of such technology.
This research project aims to develop machine-learning-based low-cost anti-poaching solutions. It presents three case studies, each exploring a unique sensor modality and use-case on the savannah, while maintaining a common framework of machine learning algorithms.

1. The first use case is the detection of gunshots using microphone-based systems.

2. The second use case is the detection of elephant footsteps using geophone-based systems.

3. The third use case is the detection of perimeter fence intrusions using accelerometer-based systems.

Each sensor modality is accompanied by a decision-tree algorithm that aims to optimize the Probability of Detection (POD)-to-False Positive Rate (FPR) ratio. Decision trees are computationally low-cost and intuitive machine learning models that are suitable for systems with low power and memory. The data used in this thesis was collected by the authors at a wildlife sanctuary in South Africa.
The project established high-performing baselines that achieved favorable POD to FPR ratios across two out of three cases. Additionally, the project provided the complete dataset and all machine learning results, including the selected features, the most effective feature selection methods, and an extensive analysis informed by local knowledge from interviewed rangers. This comprehensive approach aids the future professional development of a practically useful system tailored for the savannah environment.

Acknowledgements

First and foremost, we are extremely grateful to our supervisor Kristian Soltesz, for all his support - academic, mental and motivational.
This thesis would not have been possible without Fredrik Gustafsson, who organised the cooperation with local partners, and always provided interesting ideas. Thank you for the opportunity to do this project.
Thank you to Björn Persson, who works incredibly hard to draw attention to the beautiful wildlife we risk losing on this planet and with whom we would love to have another drink sometime soon.
Finally, we have an immense amount of gratitude to the people that we met in South Africa. Not only did you help us in many ways with our research, participating in data collection and providing valuable insights into the issues at hand - but you also made life on-site an unforgettable experience. We are happy to call several of you our close friends, and thankful to have met every single one of those that passed through during our stay.
Dankie!

Contents

1. Introduction **11**
- 1.1 Background . . . 11
- 1.2 Aim of the Study . . . 13
- 1.3 Research Questions . . . 14
- 1.4 Contributions . . . 14

2. Theory **16**
- 2.1 Machine Learning . . . 16
- 2.2 Supervised Learning . . . 16
- 2.3 Balancing . . . 17
- 2.4 Metrics . . . 18
- 2.5 Decision Trees . . . 20
- 2.6 Hard Negative Mining . . . 23
- 2.7 Feature Selection . . . 23

3. Methodology **30**
- 3.1 Hardware development . . . 30
- 3.2 Software development . . . 30
- 3.3 Testing and data collection . . . 31
- 3.4 Interviews . . . 32

4. Case 1: Gunshot detection **35**
- 4.1 Background . . . 35
- 4.2 Literature review . . . 36
- 4.3 Interviews with rangers . . . 39
- 4.4 Hardware and data collection . . . 41
- 4.5 Implementation . . . 48
- 4.6 Results . . . 52
- 4.7 Analysis . . . 59

5. Case 2: Elephant detection **70**
- 5.1 Background . . . 70
- 5.2 Literature Review . . . 70
- 5.3 Interviews with rangers . . . 73

5.4 Problem definition . . . 74
5.5 Hardware concept . . . 75
5.6 Hardware . . . 77
5.7 Data . . . 80
5.8 Modeling pipeline . . . 81
5.9 Results . . . 91
5.10 Analysis . . . 96

6. Case 3: Fence Intrusion Detection 101
6.1 Background . . . 101
6.2 Literature review . . . 102
6.3 Interviews with rangers . . . 104
6.4 Hardware and data collection . . . 105
6.5 Implementation . . . 112
6.6 Results . . . 115
6.7 Analysis . . . 118

7. Discussion 125
7.1 Model performance . . . 125
7.2 Discussion . . . 125
7.3 Future Work . . . 127

A. Code functionality in Case 2 128

B. Feature correlation for Case 2 129

C. Geophone and camera hardware problems 130
C.1 Geophone . . . 130
C.2 Camera . . . 131

D. Geophone datasets and their locations 133

E. Reconstruction of elephant footstep PSD from MFCCs 138

F. Full interviews 139
F.1 Gunshot Detection . . . 139
F.2 Footstep detection . . . 144
F.3 Fence intrusion detection . . . 147

G. Software Pipeline 153

H. Parallel projects 155
H.1 Introduction . . . 155
H.2 Weather station . . . 155
H.3 Scanners . . . 157
H.4 Lodge fence electricity . . . 159

I. Additional results 160
I.1 Gunshot detection . . . 160
I.2 Fence detection . . . 165
I.3 ROC-AUC . . . 175

J. Images taken of work at Makulu **177**
K. Laptop Hardware Specifications **180**
L. Evidence of poacher breaching fence at Makulu **181**
Bibliography **183**

1

Introduction

1.1 Background

The use of technology to address complex issues in developing regions has become increasingly prominent. From environmental conservation to public health, advanced technological tools have the potential to provide real-time monitoring, predictive analytics, and data-driven insights that support sustainable development. In the context of wildlife conservation, combating poaching has emerged as a significant application of such technology, where Anti-Poaching Systems (APS) aim to safeguard endangered species through various innovative means. Deploying these technologies effectively in remote and resource-limited environments presents unique challenges that demand robust and adaptable solutions. Among the species that APS aim to protect are elephants and rhinos, such as those pictured in figure 1.1, which are particularly targeted by poachers due to the high value of their tusks and horns.



(a) Elephants at Amboseli National Park, Kenya



(b) Rhinos at Solio game Reserve, Kenya

Figure 1.1 Photos of elephants (left) and white rhinos (right), taken by Björn Persson, Chairman of Smart Savannahs

In "Poaching Detection Technologies - A Survey" Kamminga et al [1] highlight a range of modern antipoaching technologies, along with their use cases and challenges. Existing APS technologies encompass a range of methods that operate on the ground, aerially, or even directly on animals. Examples include seismic sensors, which detect vibrations associated with movement; acoustic sensors that capture sounds like gunshots; and motion detection systems that can be used for classifying intrusions and perimeter breaches. Aerial drones equipped with thermal cameras are used to track and identify intruders, while animal-mounted GPS collars and RFID tags allow real-time location tracking and distress alerts. Each of these methods has distinct advantages and limitations. For instance, ground-based seismic sensors provide high stealth but are highly sensitive to soil composition, while thermal imaging from drones is limited by battery life and high costs.

One of the primary challenges identified by Kamminga et al is balancing sensitivity with reliability to minimize false alarms, also called false positives. High sensitivity can lead to frequent false positives, which strain resources and reduce the effectiveness of ranger responses. Ensuring a high Probability of Detection (POD) while maintaining a low False Positive Rate (FPR) is an ongoing area of research, as even robust APS solutions can suffer from environmental interferences or wildlife interaction with the equipment. For instance, false positives can arise from non-poaching sources, such as animals triggering seismic sensors or thunder triggering acoustic sensors. The paper emphasizes that achieving reliable APS with strong POD-to-FPR ratios is essential for supporting rangers in combating poaching threats and protecting endangered wildlife.

This research project aims to develop anti-poaching and conservation solutions that optimize the POD-to-FPR ratio. We hope to lay the foundational building blocks and demonstrate the potential of several technologies, so that future researchers can further the work. This research project is done in direct cooperation with the two stakeholders Qulinda and Smart Savannahs.

Qulinda [2] (the Swahili word for protect) is a company based in Linköping, Sweden, which develops and sells products used for animal supervision. Qulinda conducts testing and product development both in Sweden and in various African countries. To a large extent, Qulinda has taken over the operations of "Project Ngulia"—an informal organization based at Linköping University (LiU), where Project Ngulia continues to focus on research and prototype development, funded by Security Link (a strategic research area in societal security supported by the Swedish government) and Agtech Sweden (agricultural technology funded by the Swedish Innovation Agency). Qulinda's role is to bring this research to market by transforming it into products. Project Ngulia has had or currently maintains partnerships with the following organizations: Kenya Wildlife Service (Kenya), Airtel (Kenya), Stimson Center (USA), Clinton Global Initiative (USA), University of Nairobi (Kenya),

University of Johannesburg (South Africa), and University of Venda (South Africa), with the last two collaborations facilitated by SASUF (South Africa Sweden University Forum).
Smart Savannahs [3] is a non-profit organisation which funds the implementation and use of Qulinda's hardware in three wildlife reserves: Makulu (South Africa), Zimanga (South Africa) and Solio (Kenya). Smart Savannahs aims to fund and promote the development of smart technology which can be used in anti-poaching and animal-welfare contexts around Africa.

This research project consists of three case studies, two of which have been previously explored to varying extents by Qulinda [4, 5], and one of which investigates a new sensor modality and use case not previously addressed by Qulinda. The three projects separately explore a unique sensor modality and use case on the Savannah, while maintaining a common framework of machine learning algorithms.

1. The first use case is gunshot detection using microphone-based systems.
2. The second use case is the detection of elephant footsteps using geophone-based systems.
3. The third use case is the detection of perimeter fence breaching using accelerometer-based systems.

Each sensor modality is accompanied by a decision-tree based algorithm that performs feature extraction and classification of the sensor data. Decision trees are computationally low-cost and intuitive machine learning models that are suitable for systems on chip (SoCs) with low power and memory. The data used in this thesis has been collected by the authors at Makulu Sanctuary in South Africa.
This thesis work consists of three main structural parts, one for each tackled problem. Each part has its own sections that describe background, literature, interviews with rangers, implementation and results.

1.2 Aim of the Study

This thesis focuses on achieving high-performing baselines in each use case: gunshot detection, elephant footstep-detection and fence intrusion detection. We pose each problem as a binary classification problem. The main focus regarding the performance of the models is set to minimizing the number of false positives while maintaining good classification performance. During the course of the thesis, many areas of further development have been identified, including multiclass-classification, hardware development and communication pipelines. However, to achieve manageable results within the time-frame allocated and produce a consistent academic report, the focus has remained on binary classification. The hope of

the authors is that this research project will inspire and enable future researchers to further the work. For this purpose, the collected dataset is made public and interviews with rangers regarding the need of the end user are presented in the report. Note that the name of the test-site as well as interview persons has been redacted to various degrees to preserve the privacy of said parties.

1.3 Research Questions

1. How effective are decision trees in detecting gunshots based on microphone data collected at the savannah?
2. How effective are decision trees in detecting elephant footsteps based on geophone data collected at the savannah?
3. How effective are decision trees in detecting fence intrusions based on accelerometer data collected at the savannah?
4. What are adequate features for detection in each area of interest?
5. What are the key limitations of using machine learning models in real-world applications for wildlife conservation, and how can these limitations be addressed through data collection or model adjustments?
6. How can we reduce the amount of false positives whilst maintaining good classification performance for the chosen classification problems on the savannah?

1.4 Contributions

As the project consists of three case studies, it was possible to clearly distribute the workload among the authors. Jakob was responsible for gunshot detection (Case 1) and fence intrusion detection (Case 3) as well as the interviews with rangers. Kasper was responsible for elephant detection (Case 2), as well as conceptualizing the transition from confusion matrix to system wide properties, and the meta models.
All in all, the project presents these contributions:

- Developed and evaluated a decision tree method for detecting gunshots on the savannah using microphone data from a mobile phone (Doogee S100) .
- Developed and evaluated a decision tree method for detecting elephant footsteps on the savannah using geophone data.
- Developed and evaluated a decision tree method for detecting fence break-ins using accelerometer data from the SparkFun OpenLog Artemis.

- Developed a problem definition, hardware concept and labeling method for geophone data.
- Published a large open-source dataset containing data from all three sensors, recorded at the savannah in South Africa, significantly increasing the available open-source data online for these research areas.

2

Theory

2.1 Machine Learning

Machine learning (ML) is a subfield of artificial intelligence (AI) that focuses on building systems that learn from and make decisions based on data. It uses algorithms and statistical models to identify patterns and relationships within datasets, enabling computers to improve their performance on tasks without explicit programming. Machine learning can be categorized into supervised, unsupervised, and reinforcement learning, depending on the type of data and feedback available. The applications of machine learning range from recommendation systems to image and sound recognition. Recently, ML and AI have gained traction in wildlife conservation, finding use in areas such as advancing remote sensing methods to monitor wildlife [6] and enhancing anti-poaching efforts through predictive analysis of animal movements [7].
For more information and resources on machine learning, we recommend Google's crash course on the subject [8]. The crash course covers most of the machine learning subfields explained below, and additional resources will be provided later in the report as necessary.
In this research project, we use decision trees, a type of supervised machine learning model known for its simplicity and explainability.

2.2 Supervised Learning

Supervised learning is a machine learning approach where models are trained on labeled data, meaning that each input has a corresponding correct output. The goal is for the model to learn the mapping between inputs and outputs so it can predict the output for new, unseen data. The training process involves minimizing the error between the predicted outputs and the true labels. Common algorithms used in supervised learning include decision trees (DT), support vector machines (SVM), and neural networks (NN), with applications ranging from classification tasks like spam detection to regression problems such as predicting housing prices.

2.3 Balancing

The need for balancing

In a highly imbalanced dataset one class significantly outweighs the other. This can for example occur if one uses audio to classify gunshots. Audio signals are constantly present in the sensor, and a very small proportion of these are truly gunshots. A model trained on an imbalanced dataset can become biased towards the majority class, causing poor performance on the minority class, as the model tends to predict the majority class more frequently. Data balancing is crucial in such cases, as it ensures that the model learns to correctly identify instances from both classes, improving generalization. By addressing the class imbalance, data balancing techniques can help avoid misleading accuracy metrics and provide a more reliable model, especially for real-world applications where the minority class might be of greater interest, such as detecting rare events like gunshots. To mitigate the effects of class imbalance, techniques like oversampling, undersampling, and Synthetic Minority Over-sampling Technique (SMOTE) are common.

Oversampling. Oversampling is a technique used to address class imbalance by increasing the number of instances in the minority class. This is typically done by duplicating existing samples until the minority class reaches a comparable size to the majority class. While oversampling ensures that the model gets enough exposure to minority class data, it may lead to overfitting on the specific samples of the minority class, as the duplicated samples don't introduce new information to the model.

Undersampling. Undersampling tackles class imbalance by reducing the number of instances in the majority class, creating a more balanced dataset. This is achieved by randomly removing samples from the majority class. While this technique can help prevent bias towards the majority class, it comes with the downside of potentially losing valuable information from the dataset.

SMOTE. The synthetic minority over-sampling technique is used to address class imbalance by generating synthetic data points for the minority class. Instead of duplicating existing instances, SMOTE creates new synthetic examples by interpolating between a minority class sample x_i and one of its nearest neighbors x_{nn}. The synthetic point x_{new} is generated along the line segment connecting x_i and x_{nn} using the following equation:

$$x_{\text{new}} = x_i + \lambda \cdot (x_{\text{nn}} - x_i)$$

where λ is a random value between 0 and 1. This method increases the diversity of the minority class by producing more varied synthetic samples, improving the model's ability to generalize. However, SMOTE assumes that the interpolated points accurately represent the data distribution, which might not always be valid in complex datasets.

2.4 Metrics

Metrics tell us about the performance of a machine learning model. Different metrics generally measure performance in very distinct manners, and each metric tells us about different aspects of the model's performance. It is important to note that different metrics often have competing characteristics, where improving one metric often can lead to a performance decline in another. Therefore it is important to use a collection of metrics to get a thorough and balanced evaluation of a model's performance.
To understand these metrics, we first define a few fundamental terms:

- True Positives (TP): The number of instances where the model correctly predicts the positive class.
- False Positives (FP): The number of instances where the model incorrectly predicts the positive class when it is actually negative.
- True Negatives (TN): The number of instances where the model correctly predicts the negative class.
- False Negatives (FN): The number of instances where the model incorrectly predicts the negative class when it is actually positive.

These terms will be used throughout the following sections to define some of the most common metrics, which will be used in this research project.

Accuracy

Accuracy describes how often a model is correct by looking at the ratio between correct guesses and all guesses. It gives an intuitive sense of the general performance of the model, but it doesn't give particular weight to any specific correct or incorrect classification; in other words, it treats all classes as equally important. In practice, focusing on accuracy as a sole metric for a model can be problematic. For example, if a dataset is imbalanced, an overfitting decision model might perform well in terms of accuracy, even if the model predicts the majority class 100% of the time.
The formula for accuracy is given by

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

This equation illustrates how accuracy combines all correct predictions (true positives and true negatives) and divides them by the total number of predictions. While accuracy is straightforward and widely used, its effectiveness as a performance measure depends on the context and the balance of the dataset.

Precision

Precision measures how often a machine learning model correctly identifies instances of the positive class, focusing specifically on the proportion of true positive predictions out of all positive predictions made (i.e., both true positives and false positives). Unlike accuracy, which treats all classes equally, precision is particularly useful when the focus is on minimizing false positives, as it ignores true negatives and false negatives. However, precision does not take into account false negatives, meaning it does not penalize the model for missed positive instances. The formula for precision is

$$\text{Precision} = \frac{TP}{TP+FP}.$$

Recall

Recall is a metric that looks at how often a machine learning model correctly identifies true positives from all the positives in the dataset. Similarly to precision, recall is good for imbalanced datasets. Inversely to precision, recall is good when the cost of a false negative is high for some reason. In medicine, recall is called sensitivity and is often used because the cost of a false negative can be very high. The formula for recall is

$$\text{Recall} = \frac{TP}{TP+FN}.$$

F1-score

The F1-score is the harmonic mean of recall and precision. It works well in imbalanced datasets and when both false negatives and false positives are important. It provides a nuanced view of a model's performance in imbalanced problems and can be applied to most situations. The F1-score is calculated as

$$\text{F1} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}.$$

False Positive Rate (FPR)

The False Positive Rate (FPR) measures the proportion of negative instances that are incorrectly classified as positive. It is calculated as

$$\text{FPR} = \frac{FP}{FP+TN},$$

where a lower FPR indicates better model performance in avoiding false alarms.

Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model. It is useful and intuitive in binary classification problems. It summarizes the counts of correct and incorrect predictions by comparing predicted labels with

actual labels. The matrix consists of four elements: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). An example confusion matrix can be found in figure 2.1.

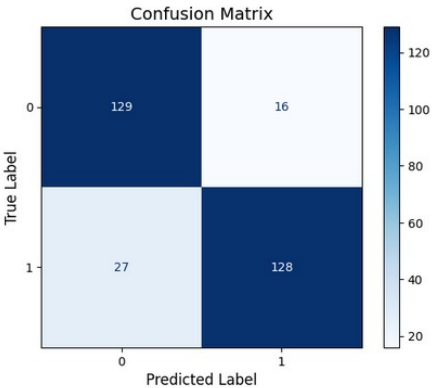


Figure 2.1 Generic confusion matrix.

2.5 Decision Trees

Decision trees are a type of non-parametric machine learning method, used for both classification and regression tasks. They are popular for their low cost and intuitiveness, and they have been selected for this work for exactly these reasons.
In essence, decision trees learn to categorize or make predictions based on a set of questions posed about the presented data. They work by recursively splitting the dataset based on feature values to create branches that lead to decisions at the leaf nodes. Decision trees resemble, unsurprisingly, trees. They begin at the base of the tree, the so-called root node, which represents the entire dataset. From the root node flows a series of decision nodes (also called internal nodes) that pose questions about the features of the data. Let's for example suppose we are looking at audio data that is either a recording of gunshot or a person speaking. In this example a question could be posed about if the frequency of the audio is higher or lower than a certain level. When we answer these questions, (yes/no), the dataset is split into subcategories. The point of the question is to try to split the dataset coming into the node into subsets that contain only one class of data (gunshot or voice). If we are able to do so, it means that we have found an indicative feature (for example a frequency that generally distinguishes voices and gunshots) that we can use to classify future data. This process occurs iteratively until enough information has been gathered and we reach the leaf nodes. The leaf nodes represent all the possible outcomes of the model. Below is a visual representation of simplified decision tree

that uses two features, duration and frequency, to distinguish between voices and gunshots.

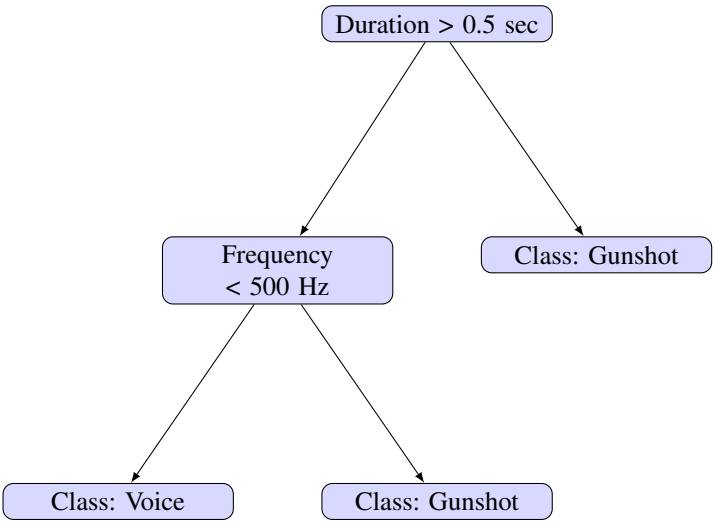


Figure 2.2 Simple Decision Tree for Classifying Audio Signals as Gunshot or Voice

Choosing Splits: Information Gain and Gini Impurity

One of the most critical aspects of decision trees is how splits are chosen at each decision node. Which question should we pose at the root node? And in which order do we pose our questions as we proceed down the tree? Intuitively, it makes sense to ask the most influential questions first, in order to gain as much information as possible as early as possible. Choosing when to ask what question is a process fundamental to how the tree works, because it directly impacts how well the data is separated and how effectively the model learns from the data. Two commonly used criteria for choosing splits are Information gain and Gini Impurity.

Information Gain

Information gain is a quantifier of how much "information" a feature provides in separating the data into distinct classes. The goal is to minimize uncertainty (entropy) with each split. A feature that reduces entropy significantly will in turn have higher information gain and make a better candidate for the split.

Information gain is calculated as the difference between the entropy of the parent

node and the weighted sum of the entropies of the child nodes:

$$IG(S,A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Where:

- S: The dataset.
- A: The feature being split.
- S_v: The subset of S where the feature A takes value v.
- $H(S)$: The entropy of the dataset S.

Entropy represents the impurity or disorder in a dataset and is calculated as:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Where p_i is the proportion of instances belonging to class i, and c is the number of classes. Information gain measures the reduction in entropy after a split, where higher values indicate a more effective split to separate classes, and lower values suggest that the split provides little information to distinguish between classes.

Gini Impurity

Gini impurity is another criterion used to evaluate splits. It measures how often a randomly chosen element from the dataset would be incorrectly classified if it were randomly assigned a label according to the distribution of class labels in the subset.

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Where:

- S: The dataset or subset of data at a given node.
- c: The number of classes in the dataset.
- p_i: The proportion of elements belonging to class i in the dataset S.

The Gini impurity ranges between 0 and 1, where 0 indicates a perfectly pure node (i.e., all elements belong to a single class), and values closer to 1 indicate a higher level of impurity or mixed class membership.

2.6 Hard Negative Mining

Hard negative mining (HNM) is a technique used in machine learning to improve model performance by reducing false positives. The core idea of HNM is to iteratively identify samples that the model incorrectly classifies as positives in the validation set and add them to the training set. These false positives, known as "hard negatives," often lie close to the decision boundary and are thus more challenging for the model to classify correctly. By systematically identifying and re-training the model on negative samples that it has previously failed at classifying, HNM helps refine the decision boundary, making it more robust and accurate. Similar to how humans may spend more time revising a difficult topic before a test, HNM revisits samples that are difficult for it classify, and by doing so becomes more robust. It makes use of a iterative training process, which entails that a validation set, separate from the training and testing set, is necessary. The main steps of HNM are as follows:

1. **Initial Training:** Train the model on the available labeled training set.
2. **Identify Hard Negatives:** Use the model to predict on a validation set, identifying the hard negative samples that were misclassified as positives.
3. **Add Hard Negatives to Training:** Add the hard negatives back into the training dataset, relabeling them as negatives.
4. **Retrain the Model:** Retrain the model on the updated training dataset, which now includes these hard negatives, to improve its discrimination between positive and negative samples.
5. Repeat the process until the number of false positives reaches an acceptable level or until additional hard negatives cease to improve model performance.

The simple idea of HNM is very effective, if one takes care to ensure that the training and validation set are representative of the test conditions and that no data leakage can occur between the sets. As always, keeping a separate test set for an unbiased estimate of generalization performance is crucial.

2.7 Feature Selection

Decision trees and other feature-based methods are extremely dependent on what features we extract from the data, as it is these features, not the raw data itself, that is used for classification. Knowing which features to include to improve classification is not a trivial process. One must first identify features that are important for the general task (eg classifying audio data) using domain knowledge. Then, for each problem and dataset, an optimal subset of features must be found. This is necessary

both to reduce computational time for making a decision as well as to make sure that non-indicative features are not used. As such, feature selection is a critical step in the machine learning pipeline, as it helps in improving model performance and reducing overfitting by selecting the most relevant features from a dataset. There are three main types of feature selection methods:

- **Filter Methods**: Evaluate the importance of each feature independently of the model. Techniques like correlation coefficients and statistical tests fall under this category.
- **Wrapper Methods**: Train the model multiple times with different subsets of features and selecting the subset that yields the best performance. This category includes methods like sequential feature selection (SFS).
- **Embedded Methods**: Perform feature selection during the model training process itself. Examples include LASSO regression and decision tree-based feature importance.

This research project will investigate both filter methods and wrapper methods to identify the optimal subset of features for minimizing the false positive rate (FPR). Specifically, we will evaluate one filter method, feature importance (as derived from decision trees), which ranks features based on their contribution to splitting nodes in the tree. Additionally, we will explore two wrapper methods: sequential feature selection, which iteratively adds or removes features to optimize performance, and Bayesian optimization, which systematically searches for the best feature subset.

Feature Importance

Feature importance is a filter-based method for feature selection that utilizes the inherent workings of decision trees to rank features based on their contribution to the model's predictive performance. In decision trees, splits are made at nodes by selecting the feature that maximizes a predefined splitting criterion, such as the reduction in Gini impurity or the gain in information. The importance of a feature is determined by aggregating the contribution of that feature across all nodes in the tree where it is used.
Mathematically, the importance score of a feature f in a decision tree is computed as:

$$\text{Importance}(f) = \sum_{n \in \text{nodes}} \Delta C_n \cdot I(f \text{ is used at } n)$$

where:

- ΔC_n is the reduction in the splitting criterion (e.g., Gini impurity or entropy) at node n.

- $I(f \text{ is used at } n)$ is an indicator function that is 1 if feature f is used for splitting at node n, and 0 otherwise.

For example, consider a dataset of audio recordings where the task is to classify whether a given recording contains a gunshot. The dataset includes the following intuitive features:

- Peak loudness: how loud the recording is at its loudest point.
- Duration of the loudest sound: how long the peak sound lasts.
- Time to peak loudness: how quickly the loudest sound occurs after the recording starts.
- Harmonics to noise ratio: a complex feature that measures the ratio of how 'harmonic' the signal is in each time segment, relative to noise.

An example decision tree trained on this dataset may compute the following feature importance scores:

Feature	**Feature Importance**
Peak loudness	0.60
Duration of loudest sound	0.25
Time to peak loudness	0.15
Harmonic-to-noise ratio (complex)	0.05

Table 2.1 Illustrative example feature importance scores for gunshot classification.

Based on these theoretical example scores, we observe that peak loudness is the most significant feature for identifying gunshots, while harmonic-to-noise ratio has very low importance. Using feature importance, we can effectively cut off unimportant features like temporal harmonic ratio to reduce computational overhead and the feature space. This ensures the model focuses only on the most relevant features, improving efficiency and interpretability. However, feature importance does not take into account complex feature interactions and non-linear relations. In this research project, we make use of feature importance as one of our feature selection methods due to its simplicity and effectiveness. For more information on feature importance, we refer to [9].

Sequential Feature Selection

Sequential feature selection is a popular wrapper method used to identify the optimal subset of features for a model. The algorithm works by adding or removing features one by one based on their impact on the model's performance, using a predefined performance metric like accuracy or precision. There are two main approaches to SFS:

- **Forward Selection**: Starts with an empty set of features and adds one feature at a time, selecting the feature that most improves the model's performance.
- **Backward Selection**: Starts with all features and removes one feature at a time, removing the feature that least affects performance.

At each step, SFS evaluates the performance of the model using k-fold cross-validation (CV):

$$\text{CV} = \frac{1}{k} \sum_{i=1}^{k} \text{performance}_i$$

where k is the number of folds, and performance_i is the chosen performance metric (e.g., accuracy or precision) obtained from the i-th fold.
SFS stops adding or removing features when the improvement in performance falls below a certain threshold:

$$\Delta\text{performance} < \varepsilon$$

where ε is a small positive value representing the threshold for stopping.
The primary advantage of SFS is its ability to directly optimize the selected performance metric, making it applicable to a wide range of models. However, it can be computationally expensive, especially when dealing with a large number of features or using backward selection. Moreover, SFS is a greedy algorithm that may not always identify the best subset of features as it is prone to get stuck in local optima if the feature space is large. For this research project we use sequential feature selection as a feature selection method due to it's popularity and ease of use. For more information on SFS, we refer to [10].

Bayesian Optimization

Bayesian Optimization (BO) is a global optimization algorithm that can be used to identify the optimal subset of features for a model. Unlike traditional methods such as grid search or random search, BO leverages a probabilistic model to guide the search process. [11] found that it performs better than random search, which is known to perform better than grid search. It also runs faster, requiring fewer iterations to find good hyperparameters. It balances exploration of new feature combinations with exploitation of promising subsets, making it an efficient and systematic approach to feature selection, particularly in high-dimensional spaces. In the case of this research project, we are utilizing decision trees, which involve numerous hyperparameters and potential combinations of features. Furthermore this project was generally carried out on restricted computational resources. For these reasons, BO stands out as a robust and computationally economic method for both feature selection and hyperparameter optimization. We also employ it due to it's global optimization nature compared to SFS and feature importance.

While BO is well-studied in the context of hyperparameter optimization, its application to feature selection provides a powerful way to refine the input to the model. By first using BO to optimize the feature subset and then applying it to tune hyperparameters, it is possible to enhance model performance while reducing computational complexity. However, BO assumes a smooth objective function, which may not always hold true, particularly in noisy or irregular datasets. Despite this limitation, BO is a robust method for addressing both feature selection and hyperparameter optimization.
Unlike grid search or random search, bayesian optimization uses information from previous evaluations to make decisions. Similar to how a human might structure a search, BO begins by testing an initial set of parameters, then modifies one or more parameters and evaluates the results. It compares the outcomes of each evaluation to determine whether performance is improving, guiding the search process toward an optimal solution.
The core components of BO include:

- Surrogate Model: A probabilistic model that approximates the objective function.
- Acquisition Function: A function that decides where to evaluate the next point based on the surrogate model.
- Exploration-Exploitation Trade-off: The acquisition function balances between exploring areas of uncertainty and exploiting areas where the model performs well.

In a linear problem, we can optimize for a certain metric, which we call the objective function, by simply computing the global minimum using for example gradient descent. However in hyperparameter tuning and other black box problems, the probability distribution of the objective function is unknown. To help, BO builds a probability model of the objective function and uses it to select hyperparameters to evaluate, using the true objective function. If we call objective function score L and the hyperparameters θ, we can describe the probability representation of the objective function as:

$$P(L \mid \theta)$$

In order to select which new set of parameters to try, BO builds a so-called acquisition function. There exists several types of acquisition functions, and they all utilize the probability representation of the objective function in order to select what new set of parameters to test. The most common acquisition function is called Expected Improvement (EI):

$$\mathrm{EI}(\mathbf{x}) = \mathbb{E}\left[\max(0, f_{\text{best}} - f(\mathbf{x}))\right]$$

Expected improvement selects new parameters by estimating the potential improvement we could gain from testing at various points in the hyperparameter space. It considers both how well the probabilistic model predicts the outcome, by looking at the mean at that point, and the uncertainty in those predictions, by looking at the variance at that point. These two elements in how EI chooses a hyperparameter set is called exploitation and exploration. EI favors testing points that either have a high chance of improving the current best result or offer valuable learning due to high uncertainty. Other acquisition functions include Lower Confidence Bound (LCB) and Probability of Improvement (PI).
This process is visually illustrated in Figure 2.3, which demonstrates how the surrogate model and the acquisition function interact to guide the optimization.

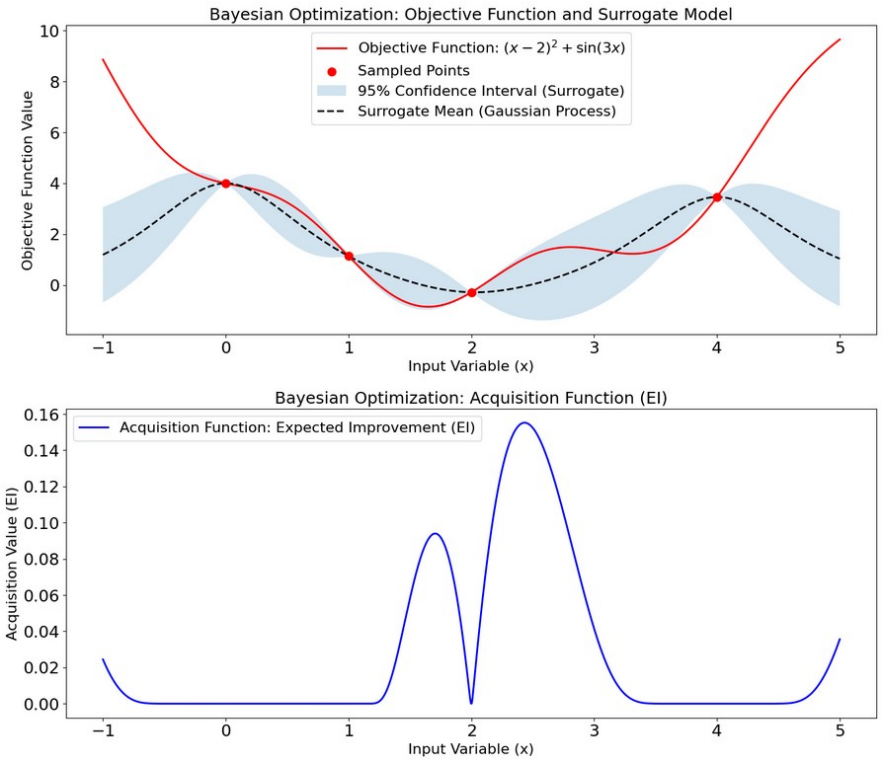


Figure 2.3 Visualization of bayesian optimization for an example. The top subplot shows the true objective function $(x-2)^2 + \sin(3x)$ (red solid line), the sampled points (red dots), and the surrogate model's mean prediction (dashed black line) with its 95% confidence interval (shaded region). The bottom subplot displays the acquisition function EI, which guides the optimization by balancing exploration and exploitation. The plots illustrate how bayesian optimization uses a surrogate model to estimate the objective function and selects the next point to evaluate based on the EI.

More information on bayesian optimization can be found in [12, 13].

3

Methodology

3.1 Hardware development

Elephant detection

For case 2, it was necessary to develop specialized hardware. A first prototype was developed on an Arduino board running Arduino C, and a second prototype was developed on a Raspberry Pi Pico running first MicroPython, and then Arduino C. The prototypes were built on prototyping boards, with the additional components modules that were soldered to the board. The final implementation is an ESP32-based LilyGo unit running Arduino C.

3.2 Software development

Gunshot detection and fence intrusion detection

The software for data processing, model development, and fine-tuning was developed using Python 3.12. We chose Python for its extensive libraries and community support in scientific computing and machine learning. All code was written in Jupyter Notebook format, initially selected for the ease of uploading to GPU clusters and the ability to train models on cloud services like Kaggle or AWS. As the project progressed, we continued using Jupyter Notebooks due to their inherent flexibility and rapid development capabilities.
The codebase extensively utilizes the scientific computing package NumPy, along with data processing libraries such as SciPy, scikit-learn (sklearn), and imbalanced-learn (imblearn). For model development, metric implementation, and analysis, we employed tools like sklearn, Seaborn, SHAP, and MLflow. MLflow is a machine learning experiment tracking tool used for managing the lifecycle of machine learning projects. While its usefulness is highlighted when tracking learning curves of deep learning models, its logging capabilities nonetheless provide a standardized way to log results and parameters even for simple models such as decision trees. Optimization tasks relied on libraries such as sklearn, scikit-optimize (skopt), and Hyperopt.

To organize our workflow, training parameters, data directories, and other configurable settings that might change during experimentation were stored in a config.yaml file. Using a YAML configuration file allowed us to manage these parameters efficiently and maintain consistency across different runs.
All raw and processed data were stored in Dropbox, which facilitated easy access and synchronization across various development environments and team members.
The software pipeline followed a structured workflow:

1. Data Collection and Labeling: The workflow began with collecting data, which was labeled automatically for passive data or semi-automatically for active data. This labeled data was then split into training, validation, and testing datasets.

2. Feature Selection and Initial Results: A comprehensive set of features was selected, and initial results were obtained using Hard Negative Mining (HNM) with the full feature set. We applied three feature selection methods—feature importance, sequential feature selection, and Bayesian optimization—each yielding a subset of features. HNM was used with each subset to improve model performance, and results were recorded for comparison.

3. Hyperparameter Optimization and Final Evaluation: The best-performing feature subset underwent hyperparameter optimization using Bayesian optimization. An evaluation was made as to whether or not this improved performance on unseen data.

A visual overview of the software pipeline, illustrating these steps, can be found in Appendix G.

Elephant detection

The data processing and modeling for case 2 was written in Python. Some code was first written in Matlab and then translated, as Matlab has very useful plotting and debugging functions. The main libraries employed were NumPy, Pandas, SciPy, Scikit, Imbalanced Learn, Seaborn, and MLFlow. Furthermore, Optuna and XGBoost were used. The model settings were stored in a YAML file. The data was stored on Google Drive. The modeling pipeline is described in detail in Section 5.8.

3.3 Testing and data collection

Data collection was carried out at Makulu, a wildlife refuge located in the Limpopo province of South Africa covering an area of slightly less than 30 square kilometers hectares. Makulu has an ongoing cooperation with SmartSavannahs and Qulinda. SmartSavannahs and Qulinda (described in more detail in 1.1) provide Makulu with hardware suitable for antipoaching and animal supervision, such as smart cameras

and tracking collars. It is thanks to the good relations between these actors and Makulu that this work was possible to carry out. The fieldwork at Makulu was largely funded by the Minor Field Studies (MFS) grant, funded by the Swedish International Development Cooperation Agency (SIDA) [14]. Makulu is located in a savannah grassland biome. The savannah is characterized by widely spaced trees that do not form a closed canopy [15]. Between trees, tight bushes rize one to several metres above the ground. Figure 3.1 illustrates the vegetation present during data collection. Note that the clear area in the picture was created by the sanctuary managers and such areas are relatively rare. The area is mostly flat, but has a few ridges and hills (known as koppies) with a topographic prominence of up to 250 m.



Figure 3.1 Example of the vegetation present in the savannah biome of Makulu

3.4 Interviews

To ensure that the outcomes of this research project effectively support rangers and wildlife sanctuaries, on-site interviews were conducted to gather their insights on the most important features and desired functionalities of such technology. Recognizing that neither purely statistical models nor ranger observations alone can fully capture on-the-ground realities [16], the interviews followed a semi-structured

format [17], combining prepared questions with open-ended discussions. This approach provided interviewees the flexibility to highlight topics they considered significant while ensuring that key areas of interest were systematically addressed.
In a semi-structured interview, the interviewer follows a general outline of questions but allows the conversation to flow naturally. This approach is especially useful in exploratory research, as it can reveal nuanced insights and unanticipated perspectives that might be missed in a rigidly structured format.
The semi-structured format was chosen for this study to encourage rangers to express their opinions openly while also ensuring that the interviews covered relevant topics related to technology implementation, usability, and practical requirements.
The following points of interest and questions were prepared in advance in the interviews, where X stands for the three technologies investigated in this thesis:

- Have you worked with or been exposed to X sensor system before? How was your experience?
- What would you say is the most important part of X sensor system?
- What could be some of the problems that X sensor system faces on the savannah?
- What are some strategies used by poachers, related to X sensor system?
- What are some important aspects to think about when it comes to the placement of X sensor system?
- If you could wish for something, an ability or function, from X sensor system, what would that be?
- Is there anything more you would like to mention or for us to think about when implementing X sensor system?

The following persons were available during the field study and also deemed knowledgeable about the needs of the user. Note that full surnames are not provided due to privacy reasons.

Name	Age	Occupation	Knows Most About
Tiaan A.	21	Game Ranch Manager. Assistant Sanctuary Manager at Makulu	Anti-poaching
Junel D.	22	Ranger, Volunteer Coordinator at Makulu	Animal welfare and behavior
Didi S.	48	Chairman and Managing Director at Makulu	Business
Lucas G.	32	Sanctuary Manager	Anti-poaching, animal welfare, farm management

Table 3.1 Interview Subject Information

4

Case 1: Gunshot detection

4.1 Background

The illegal wildlife trade is a highly lucrative and destructive industry, often ranking alongside the drug and arms trades in both economic value and severity. The trade is estimated to generate billions of dollars annually, with devastating consequences for wildlife populations and ecosystems [18].

There are several reasons why poaching exists. Two of the most prevalent types can be classified as meat poaching and luxury poaching. Meat poaching arises out of necessity and survival, without discrimination regarding the species targeted. In many rural and remote communities, bushmeat is one of the most accessible and affordable sources of protein. As a result, poachers often target wildlife such as antelopes, monkeys, and giraffes, killing these animals illegally to feed their families or to sell at local markets. This form of poaching is primarily driven by poverty and a lack of alternative livelihood options, making it as much a socio-economic issue as it is a wildlife conservation challenge. Luxury-poaching, on the other hand, is driven by the lucrative black market for animal parts that are considered status symbols or hold cultural significance. Rhino horn, for instance, is mistakenly believed to have medicinal properties and is used in some cultures as a remedy for fever or as a purported cancer treatment [1]. Elephant ivory is prized for its aesthetic value and is often carved into jewelry or ornaments. The demand for these products comes mainly from affluent buyers in international markets, where rhino horn can sell for as much as 17,000 USD per kilogram, and elephant ivory can command prices of up to 600 USD per kilogram [19, 20]. The high prices associated with these animal parts make luxury-poaching a highly profitable activity, often attracting organized and well-funded criminal networks that are willing to go to great lengths to obtain and smuggle these goods.

[1] Rhino horn is primarily composed of keratin, the same protein found in human hair and nails. Studies have found no pharmacological properties that would support claims of it treating fever, cancer, or any other illness.

South Africa is one of the most important countries in the world when it comes to both rhino populations and poaching activities. Home to nearly 80 percent of the world's rhino population [21], the country has become a critical stronghold for rhino conservation. However, this makes it a prime target for poachers seeking to exploit the high demand for rhino horn in international black markets.

In order to poach successfully, poachers employ a wide range of tactics and weapons depending on their goals and funding. For meat-poaching, where local hunters are primarily targeting wildlife for food, rudimentary low-calibre firearms are commonly used (see section 4.3). These weapons are less expensive and more accessible. In contrast, luxury-poaching operations often involve high-caliber rifles, tranquilizer guns, and even night vision equipment.

Efforts to combat poaching have increasingly turned to advanced technological solutions, including the use of acoustic sensors to detect gunshots in wildlife reserves. By placing strategically positioned acoustic sensors throughout protected areas, conservationists can monitor for the distinct sound signatures of gunshots and respond rapidly to poaching events. This approach allows for real-time alerts to park rangers and other enforcement agencies, enabling quicker mobilization and increasing the chances of intercepting poachers before they can flee the scene.

4.2 Literature review

In the context of our research, reviewing relevant literature is essential to ensure a robust foundation and informed decision-making. The selected papers were chosen based on their direct relevance to our area of focus: Machine learning for gunshot detection. The chosen papers provide insights into methodologies, frameworks, and applications similar to our own. By critically analyzing these studies, we aim to identify best practices and recognize potential pitfalls.

Machine Learning Analysis on Gunshot Recognition

In this paper, the authors explore a range of machine learning models for recognizing gunshot sounds from pistols and rifles [22]. They note that most research in gunshot classification relies on recordings available on YouTube or sound effect libraries. The authors contribute a data synthesis method based on environmental geometry. They use 19 features, including Mel-frequency cepstral coefficients (MFCCs) with 13 coefficients, spectral spread, spectral flatness, spectral entropy, spectral skewness, harmonic ratio, and kurtosis. The paper discusses three types of models commonly used for gunshot classification: ensemble classifiers, support vector machines, and neural networks. For ensemble classifiers, they consider a decision tree as the base model and highlight bagging, boosting, and stacking as the main variations of ensemble learning.

For feature selection, the authors employ the wrapper-type method Sequential Feature Selection and select the following features: MFCC coefficients (1, 3, 4, 6, 7, 8, 13), spectral skewness, and kurtosis.
For hyperparameter tuning, the authors use a Bayesian optimization algorithm with 30 iterations. The hyperparameter search range for ensemble classifiers includes the following: ensemble method = bagging, AdaBoost, RUSBoost; number of learners = 10–500; learning rate = 0.001–1; and maximum number of splits = 1–29,039. The optimized parameters yielded the AdaBoost ensemble method, with 14 learners, a learning rate of 0.8793, and a maximum number of splits of 145.
The authors achieve high accuracy across all discussed models, demonstrating the robustness of the models under varying levels of additive white Gaussian noise applied to the test set.

Gunshots Localization and Classification Model Based on Wind Noise Sensitivity Analysis Using Extreme Learning Machine

This paper addresses the challenges faced in gunshot detection and localization under windy conditions[23]. The research utilizes the Airborne Gunshots Dataset, an online dataset of gunshots recorded in an unspecified acoustic environment, along with simulated windy conditions. In terms of classification, the paper evaluates the accuracy of various machine learning (ML) and deep learning (DL) classifiers across different wind noise signal-to-noise ratios (SNRs). It also compares the performance of conventional filters versus adaptive filters for wind noise removal on the dataset.
The study employs eight classes of gunshots and uses SMOTE to balance the training dataset. The authors assess the accuracy of several ML classifiers, including Extreme Learning Machine (ELM), Support Vector Machine (SVM), Naive Bayes (NB), k-Nearest Neighbour (k-NN), and Random Forest (RF). They also analyze the performance of various 1D-CNN architectures.
The authors recommend their developed Hadamard product combined with wavelet de-noising as the optimal method for achieving high SNR among all tested filtering methods and simulated wind conditions. Among traditional filters, the median filter demonstrated the best performance. ELM was the top-performing model across different wind noise simulations; however, NB and ELM exhibited classification times exceeding 4 ms, while RF achieved classification times under 1 ms on the same hardware.
The authors further recommend a hybrid feature combination of MFCC and Gammatone frequency cepstral coefficients (GFCCs) for the RF model. They conclude by emphasizing the importance of gathering training and testing data from real-world environments rather than relying on simulated datasets.

Machine learning inspired efficient acoustic gunshot detection and localization system

The paper presents a real-time gunshot detection and localization system, utilizing the Airborne Gunshot library for training, which contains audio samples from a wide variety of firearms[24]. Each sample is 3 seconds in duration, with added white Gaussian noise at various SNR values. The initial features considered before feature selection include MFCC, GTCC, Linear prediction coefficients (LPC), and spectral centroid. Feature selection is conducted by iteratively testing different feature combinations and comparing their performance. The recommended feature set comprises LPC, MFCC, and GTCC.

The authors emphasize that there are three primary sound sources associated with a gunshot: mechanical action, muzzle blast, and shock wave. Mechanical action is quieter than the other sources and is thus disregarded in their study. They note that the muzzle blast lasts approximately 3–5 ms, has a frequency range of around 700–1000 Hz, and can be audible at distances of several hundred meters to a kilometer. According to the authors, the muzzle blast is the main acoustic indicator of gunshots. Furthermore, they argue that methods relying on handcrafted features are the most reliable and computationally efficient when data is limited. It is worth noting that their dataset is relatively small, containing a total of 1,000 samples, with 250 positive and 750 negative instances.

Gunshot Detection and Direction of Arrival Estimation Using Machine Learning and Received Signal Power

This master thesis, written by David Grahn and Timothy Cooper, investigates whether it is possible to aid anti-poaching efforts in Kenya with a gunshot detection and localization using an array of microphones[4].

For gunshot detection, a range of machine learning methods are explored, including both feature based methods and a neural network. For the feature based methods, the following are investigated: Decision Trees, Naive Bayes Random forest, K-nearest neighbour and Support Vector. They utilize Sequential Feature Selection for narrowing down the feature space and for the decision tree model the following features were found most useful: 3 MFCC features, zero crossing, piecewise mean, piecewise standard deviation and peak time.

They achieve the most reliable results with the Naive Bayes model, however all models show a large amount of false positives, which the authors stress is an issue when dealing with shot detection.

Relevance of Papers

Each paper reviewed played a pivotal role in shaping our research direction and methodology. The key contributions from each paper are as follows:

1. “Machine Learning Analysis on Gunshot Recognition”

- Inspired the use of key features such as MFCCs, spectral spread, and harmonic ratio.
- Highlighted SFS as a valuable wrapper-type feature selection method.
- Brought attention to bayesian optimization for hyperparameter optimization selection.

2. “Gunshots Localization and Classification Model Based on Wind Noise Sensitivity Analysis Using Extreme Learning Machine”
   - Tackled a similar problem of wind noise reduction in gunshot recognition models.
   - Provided potential solutions, even though a custom approach was ultimately adopted.
3. “Machine Learning Inspired Efficient Acoustic Gunshot Detection and Localization System”
   - Suggested the effectiveness of short sample durations, such as two or three seconds, for gunshot recognition systems.
4. “Gunshot Detection and Direction of Arrival Estimation Using Machine Learning and Received Signal Power”
   - Addressed gunshot detection in the savannah environment, aligning with the focus of this research.
   - Highlighted the critical issue of reducing false positives for gunshot detection in this context.

4.3 Interviews with rangers

The interviews were conducted in a semi-structured format, allowing the conversation to flow naturally and enabling the interviewees to explore topics of interest in depth. The guiding questions used for the interviews are provided in Section 3.4.
The following text presents the insights gathered from interviewing four individuals working at the test site about gunshot detection. The term "shotspotter" is a common term in South Africa for gunshot detection systems, and the two terms are used interchangeably.
The full transcript can be found in F.1.

Aims and important issues

Tiaan A. provides valuable insight into how shot-spotting technology could assist rangers, drawing on his experience with a large farm that employs such systems.

He highlights the importance of reducing false alarms and improving overall efficiency: rangers, he notes, have limited time and must spend it wisely. Whenever the shot-spotter triggers due to an elephant snapping a tree branch, for instance, critical manpower is wasted. He suggests sending an audio sample of each alert to a ranger who can quickly judge whether it genuinely indicates gunfire. Equally significant is the idea of distinguishing between calibers: Tiaan observes that larger-caliber rifles tend to be linked with luxury poaching targets such as rhinos, whereas smaller calibers often indicate meat poachers. Better identification of these threats would help rangers prioritize responses based on the seriousness of each incident.

Didi S., Makulu's Chairman and Managing Director, adds a broader perspective on how such systems can benefit both security and long-term business interests. While shot detection by its nature is reactive, he believes that an extended detection range and more precise localization can nonetheless provide critical information, such as whether a shot originated from inside Makulu's boundaries or a neighboring property. He further emphasizes that poachers commonly use homemade or purchased silencers, thus underscoring the need for devices that can detect quieter or muffled shots. Didi also advocates for multipurpose systems: by designing microphones that can pick up and locate drones, one could catch poachers in the preliminary stage, when they are scouting the reserve to locate valuable game. From his viewpoint, the system's design must balance practicality and flexibility, so that rangers can tune settings with minimal specialized support, and ideally rely on an app interface that shows an alert alongside its confidence level.

Potential problems

A clear recurring theme among Tiaan, Didi, Junel D., and Lucas G. is the prevalence of false alarms and the maintenance burden. Junel has seen shot-spotters installed atop tall poles on elevated *koppies* (small hills), where they detect sound more effectively but become much more difficult to service. Helicopters may be required to transport heavy equipment for repair or calibration, placing further demands on time and resources. Similarly, Lucas recalls that a neighboring sanctuary's shot-spotters frequently produced spurious alarms whenever elephants broke branches. This not only eroded trust in the system but also wasted rangers' time, as each alert had to be checked urgently in cooperation with nearby landowners.

Environmental variability poses another challenge. Junel details how the cracks and snaps caused by elephants tearing tree bark can sometimes mimic gunfire, but typically produce secondary sounds that distinguish them from an actual shot. Building a model to handle every possible false trigger (from thunderstorms to branch-snapping) demands large volumes of diverse training data and, most likely, more advanced algorithms that can interpret sequential audio patterns. Furthermore, poachers' own ingenuity complicates detection. Many use makeshift rifles loaded in unconventional ways, or add improvised suppressors. The acoustic signature of such weapons differs substantially from factory firearms, requiring shot-spotters to ac-

count for numerous potential noise profiles. Finally, systematic maintenance of numerous sensor devices becomes especially difficult in remote, wildlife-rich settings, where travel is slow and at least two individuals are needed for safety on each excursion.

Future Goals

While shot-spotting currently functions primarily as a reactive measure, all interviewees see significant room for broadening its capabilities. Didi and Junel, in particular, envision systems that go beyond detecting *only* gunshots. Junel emphasizes that capturing sounds prior to a shot being fired—such as the loading of firearms or the idle hum of a drop-off vehicle—could offer a more proactive approach. Coupled with drone detection and accurate localization, rangers would stand a greater chance of preventing incidents before poachers find their targets.

Lucas adds that having reliable directional or triangulated data would enable rangers to reach the scene faster, thus improving the likelihood of apprehending poachers, even if shots have already been fired. In a broader sense, Tiaan and Didi both support designing these systems with adaptability in mind: each farm has its own topography, and an easily reconfigurable solution would be far more practical than one requiring major on-site recalibrations. An app-based interface that communicates in real time, indicates probabilities of detection, and clearly separates high-urgency from low-urgency alerts would reduce ranger fatigue while improving situational awareness. Looking forward, the interviewees emphasize that lowering false positives and maintaining a cost-effective deployment model will be vital to ensuring that these systems gain widespread adoption and remain useful long term. If these goals can be met, shot-spotting could advance from merely documenting illicit gunfire to playing a pivotal role in *preventing* wildlife poaching at Makulu and other reserves.

4.4 Hardware and data collection

Test-site description

Makulu is located in a savannah grassland biome. The savannah is characterized by widely spaced trees that do not form a closed canopy [15]. As a result, data collected in this environment can differ from data recorded in dense forested areas [4], where trees and vegetation are more abundant and can significantly influence acoustic propagation.

Although Makulu contains several koppies (hills), which could theoretically reflect or obstruct sound, this study did not quantify their impact on acoustic measurements.

Hardware

The hardware used for collecting audio data was a set of two Doogee S100 mobile phones. The Doogee S100 is developed for rugged conditions and long-lasting battery life, making it suitable for collecting data in all types of weather and for longer periods of time. The app "Easy Voice Recorder" found in the Android app store was used. This allowed for tuning recording settings. The recorded audio had the following specifications:

- Sample resolution: 32-bit float
- Sampling frequency: 16kHz
- Channel: Monochannel

The hardware was generally placed on sturdy trees using zip-ties at approximately 1.8 meter height above the ground. Data was collected in flat areas rather than on koppies or hilltops, due to the difficulty to access koppies on the test-site. The phone was placed with the microphone facing downwards towards the ground, as seen in figure 4.1.



Figure 4.1 Placement of microphone on tree

The red piece of cloth placed on the microphone aids with wind noise reduction and is explained in section I.1.

Passive data

For passive data, a set of common types of sounds were identified. The set consists of the following types of sounds:

- "Background noise": All common sounds heard at night such as wind, birds calling, and trees swaying.
- "Lions": Lions roaring, often observed during night at regular intervals.
- "Hyenas": Hyenas calling, often observed during night and during feeding of the rescued hyenas.
- "Storm": Rain and thunder.
- "Human activity": Vehicles passing by the microphone, humans speaking in vicinity of the microphone, machines such as generators active in the vicinity of the microphone.

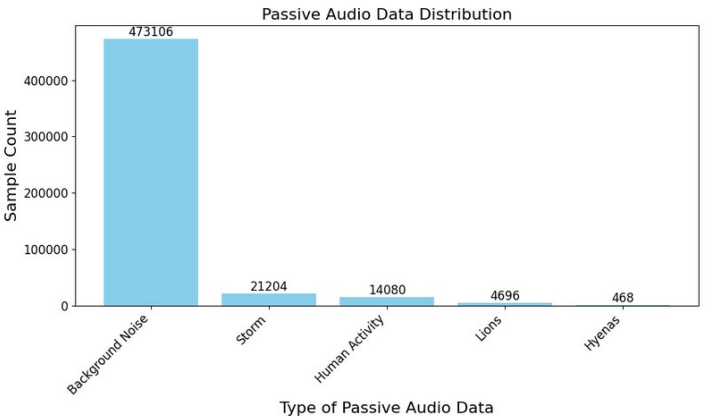


Figure 4.2 Histogram of passive audio data distribution. Each sample contains 2 seconds of audio, totaling to just below 300 hours of data.

For recording background noise, the microphones were placed near watering holes during night time in order to emulate the conditions where poaching is more likely to happen. Any sound made by animals passing by the microphones during the recording would be classified as background noise. For recording any other sort of background data, such as lions roaring, or human activity, microphones were placed in the vicinity of where such data occurs, and labeled and analysed offline.

The different waveforms of background noise can be found in figures 4.3, 4.4, 4.5, 4.6, 4.7.

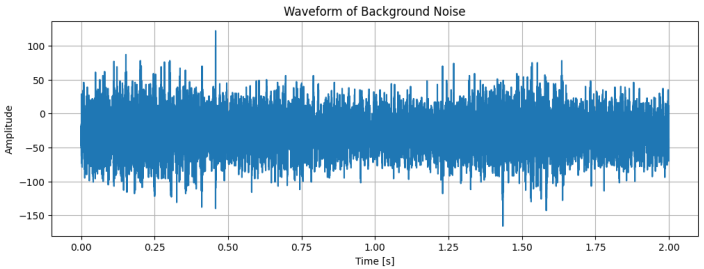


Figure 4.3 Waveform of a background noise sample with very little contents

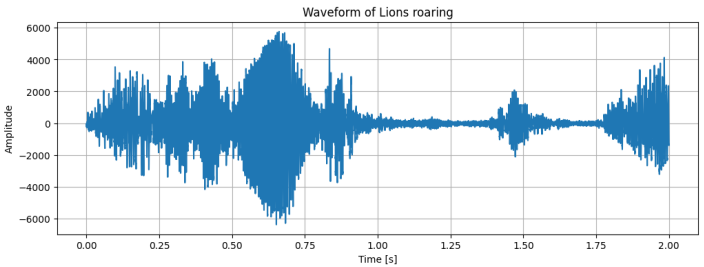


Figure 4.4 Waveform of a lions roaring

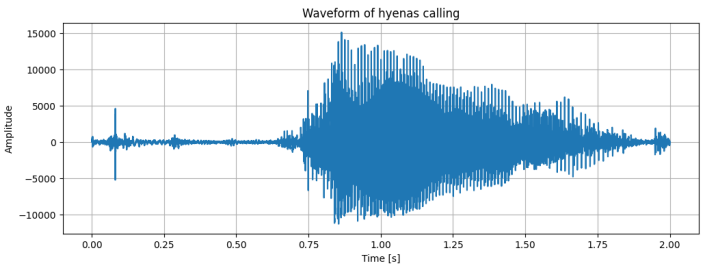


Figure 4.5 Waveform of a hyena calling

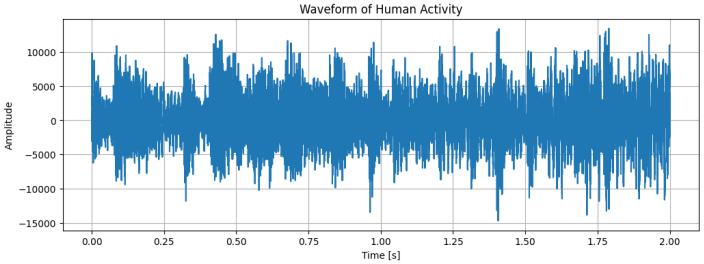


Figure 4.6 Waveform of a people speaking with car sounds in the background

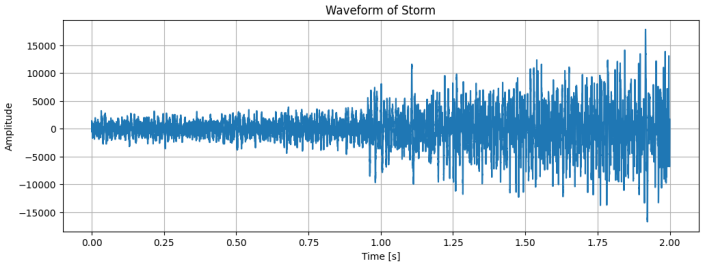


Figure 4.7 Waveform of thunder and rain sounds

Active data

For collecting active data, the data collection site was limited to a certain area inside the test-site where guns could be fired safely. The two microphones were placed at various locations around the firing site, varying in range separating the microphone and the site of fire. Three types of weapons were available during the testing , and the data was labeled in regards to both which weapon was fired, as well as the distance from the site of fire. The three weapon types were: hunting rifle (pictured in 4.9), shotgun and ArmaLite rifle (AR). A table and histogram of the sample counts by weapon type and distance can be found in table 4.1 and figure 4.8. The different waveforms of active data can be found in figures 4.10, 4.11, 4.12.

Table 4.1 Sample Counts by Weapon Type and Distance

Weapon	10m	50m	150m	350m	Total
Hunting Rifle	18	16	6	5	45
Shotgun	49	52	20	17	138
AR	49	46	38	16	149

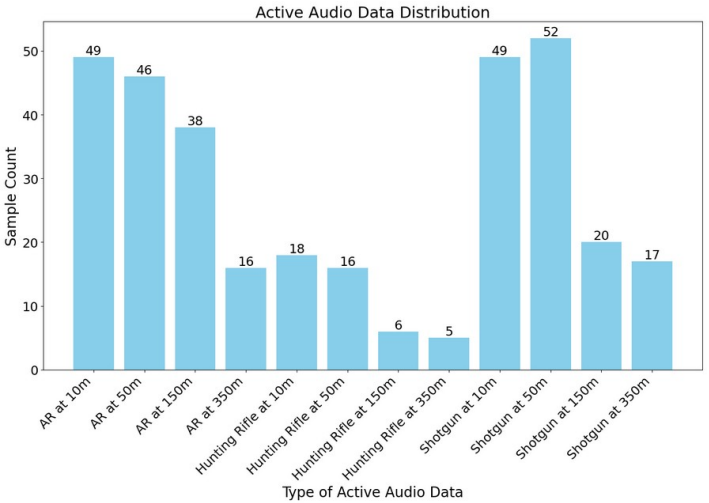


Figure 4.8 Histogram of active audio data distribution



(a) Hunting rifle used during testing, with .270 caliber.



(b) Ammunition used by hunting rifle, a .270 caliber.

Figure 4.9 Hunting rifle and corresponding ammunition used during testing.

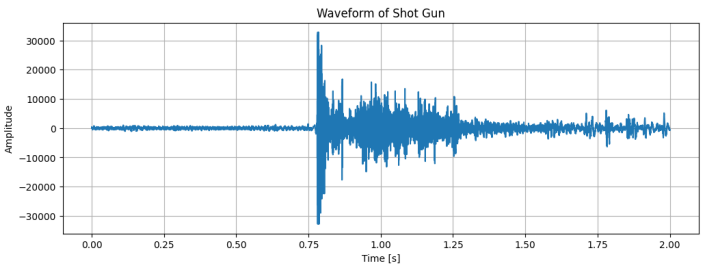


Figure 4.10 Waveform of a shotgun fired at 10 meters distance from the microphone

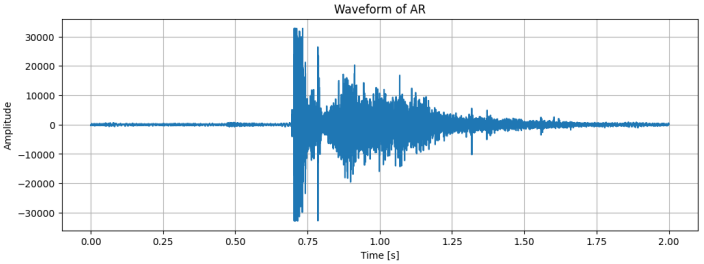


Figure 4.11 Waveform of a AR fired at 10 meters distance from the microphone

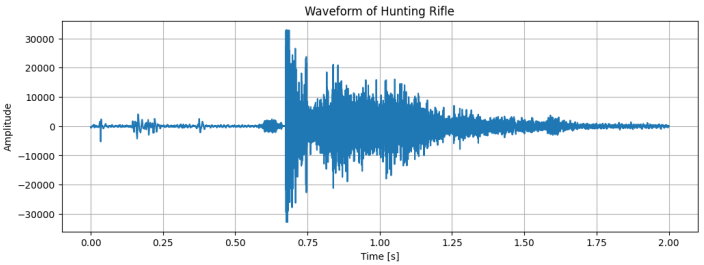


Figure 4.12 Waveform of a Hunting Rifle fired at 10 meters distance from the microphone

4.5 Implementation

Preprocessing

Window Size. Using the `librosa` library in Python, recordings were split into windows of 2 seconds with a 1 second stride. This means each window overlaps with its neighboring windows by 50%, ensuring that no information is lost between samples. All data was manually reviewed and labeled using a Python script.

Data splitting. Passive samples were split into train, test and validation splits using time as a distinguishing factor. Passive samples collected one night were used during training while passive samples collected the next night were used for testing. Active samples originating from the same true gunshot (for example two following samples that both contain clear traces of the same gunshot recording) were regarded as one individual piece of information. By doing so individual pieces of information containing up to two samples could be randomized among the splits, without risking data contamination.

Balancing. To balance the training set, SMOTE, undersampling, and oversampling were applied. During data preprocessing, the internal subclasses of passive data (e.g., background noise and storm) and active data (e.g., shotgun at 10 meters,

AR at 350 meters) were treated as separate classes to facilitate subclass balancing. SMOTE was used within the passive data subclasses to balance the different types of sounds such as thunder and hyenas. Oversampling was experimented with to balance the active data subclasses (the small number of samples means SMOTE is not possible), but results were unsatisfying and therefore it was not applied - leaving the active data unbalanced across it's internal subclasses. Across the final binary classes, SMOTE was used to balance the final sound and shot classes. The results of balancing can be seen in 4.13 and 4.14.

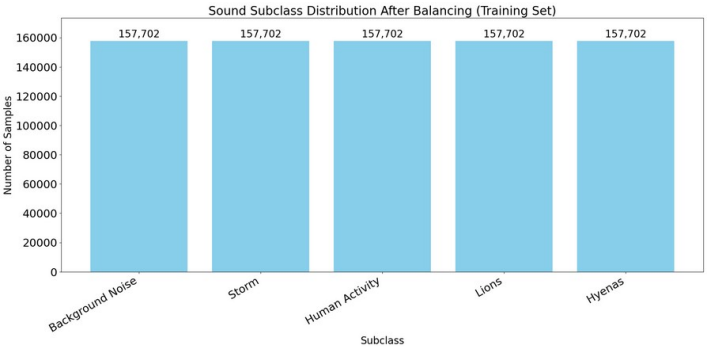


Figure 4.13 Distribution of passive audio subclasses after performing SMOTE

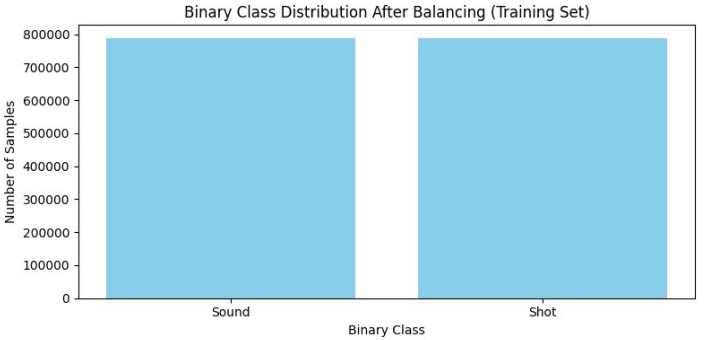


Figure 4.14 Distribution of binary audio classes after performing SMOTE

Features

The following feature suite was selected based on paper recommendations, intuition and early experimental results.

1. **Harmonic-to-Percussive Ratio (HNR)**: This feature is the ratio of the mean square amplitude of the harmonic component to that of the percussive compo-

nent. It provides a measure of the tonal versus transient qualities in the audio, where higher values indicate a more tonal (harmonic) sound and lower values a more transient (percussive) sound.

2. **High Frequency Energy Ratio**: This feature represents the ratio of energy in the high-frequency range (above 2000 Hz) to the total spectral energy. It can be useful for distinguishing sounds with significant high-frequency content, such as sharp percussive events, from those dominated by lower frequencies, like bass notes.

3. **Standard Deviation of Amplitude Envelope**: Calculated as the standard deviation of the amplitude envelope (absolute values of the waveform), this feature reflects the variability in signal amplitude over time. High values suggest dynamic variation, while low values indicate a more stable or consistent amplitude.

4. **Spectral Centroid**: The spectral centroid is the weighted mean of the frequency spectrum and is often associated with the perceived brightness of a sound. Higher spectral centroid values correspond to brighter or sharper sounds, while lower values indicate darker or duller sounds.

5. **Spectral Spread**: This feature quantifies the dispersion of the frequency spectrum around the spectral centroid, essentially measuring the spread of energy across frequencies. Sounds with a narrow frequency range will have lower spectral spread, while those with a wide frequency distribution, such as noise, will have higher values.

6. **Mean Harmonic Amplitude**: This feature is calculated as the average amplitude of the harmonic component extracted from the audio signal. Higher values indicate a stronger harmonic (tonal) structure, whereas lower values suggest a more noise-like sound.

7. **Mel-Frequency Cepstral Coefficients (MFCCs 1-13)**: MFCCs are derived from a log-power spectrum on a nonlinear mel scale, emulating human auditory perception. The first 13 coefficients capture the broad spectral shape and are commonly used in speech and audio recognition tasks. Each coefficient represents a particular aspect of the spectral envelope, with lower coefficients capturing coarse structure and higher coefficients capturing finer details.

Hard Negative Mining

Hard negative mining was applied on all of the different feature sets discussed below. The results come from applying HNM on that feature set, unless specified otherwise. The number of iterations for HNM was determined by an early stopping criterion: the process continued until there were no false positives left in the validation set. As long as false positives were present in the validation set, they

were iteratively moved to the training set. At each iteration of HNM, the confusion matrices and the metrics outlined in Section 2.4 were evaluated.
A decision tree classifier model was used without predefined hyperparameters, allowing parameters such as tree depth and criterion to be automatically selected by scikit-learn.

Feature importance

To identify and retain the most influential features, the decision tree model's built-in feature importance attribute was utilized. After training the model, the importance of each feature was printed and visualized, revealing their relative contributions to the classification process. Features contributing less than a subjective threshold of 0.1% to the decision process were deemed negligible and subsequently removed. This step aimed to streamline the feature set.

Sequential Feature Selection

Using the original training, test, and validation data splits, sequential feature selection was applied to reduce the feature space. The selected features were then fed into the HNM algorithm for testing. SFS was implemented with backward feature selection, as initial tests with forward feature selection produced inferior results.
To align with the project objective of minimizing false positives, FPR was used as the primary optimization criteria for the SFS algorithm. A decision tree classifier model was employed without predefined hyperparameters, allowing parameters such as tree depth and criterion to be selected automatically by scikit-learn.

Bayesian Optimization

Using the original training, test, and validation data splits, bayesian optimization was applied to both reduce the feature space and identify optimal hyperparameters. BO was generally performed as a two-step process. In the first step, BO optimized over a feature mask to identify the optimal feature selection. For this feature selection, a decision tree classifier model was used without predefined parameters, allowing hyperparameters such as tree depth and criterion to be chosen automatically by scikit-learn.
Once feature importance, SFS and BO had all been used to create subsets of features, the results of using those subsets were compared. Using the features that led to the best FPR, BO was applied again to search the hyperparameter space and find the optimal hyperparameters. The optimal features and hyperparameters were then fed into the HNM algorithm for testing.
The following hyperparameter search space was investigated by BO:

- **criterion**: `'gini'` or `'entropy'`
  This hyperparameter determines the function used to measure the quality of a split.

- **splitter**: `'best'` or `'random'`
  This setting controls how the algorithm chooses the split at each node. `'best'` selects the optimal split, which often yields higher accuracy, while `'random'` chooses a split randomly among the top candidates, potentially speeding up the process.

- **max_depth**: Integer between 1 and 20
  This parameter sets the maximum depth of the tree. Limiting the depth prevents overfitting by controlling the complexity of the tree. Lower values create simpler models, while higher values allow more complex trees that may capture more details but risk overfitting.

- **min_samples_split**: Integer between 2 and 20
  This is the minimum number of samples required to split an internal node. Higher values prevent the model from learning overly specific patterns by making splits only when a node contains a sufficient number of samples.

- **min_samples_leaf**: Integer between 1 and 20
  The minimum number of samples required to be at a leaf node. Setting a higher value can smooth the model, reducing variance and preventing the creation of nodes that do not contribute much predictive power.

- **max_features**: `'sqrt'`, `'log2'`, or a float between 0.1 and 1.0 representing a fraction of the number of features
  This parameter defines the number of features to consider when looking for the best split. Using a fraction or subset of features can help to reduce overfitting and speed up computation.`'sqrt'` and `'log2'` use subsets that can create more diverse trees.

4.6 Results

In this section, a range of results are presented. The results for each model include a confusion matrix and a set of quantitative metrics. For a refresher on the definition of the various metrics, we refer to Section 2.4.
In this case, the positive class is gunshots, consisting of the data explained in Active data (4.4) while the negative class is other noise, consisting of the data explained in Passive data (4.4).

Base model

The base model makes use of all available features. Using all available features does not take into account computational complexity nor the effects of a large feature space on performance, but can give a rough performance estimate.



Figure 4.15 Confusion Matrix for gunshot detection using all features with HNM for three iterations

Metric	Value
Accuracy	0.9997
Precision	0.9315
Recall	0.5667
F1-Score	0.7047

Table 4.2 Quantitative metrics for gunshot detection using all features with HNM for three iterations

Feature importance

Plotting the feature importance of the gunshot detection model yields the following:

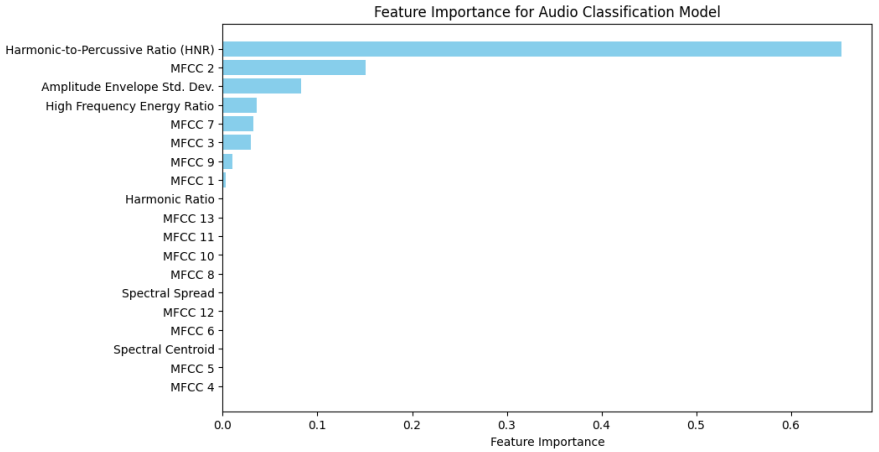


Figure 4.16 Plot of feature importance from model used in 4.6

By cutting off features with less than 0.001 importance, in other words features that contribute to less than 0.1 % of the decision-making capabilities, the remaining features become:

1. Harmonic-to-Percussive Ratio (HNR)

2. Amplitude Envelope Standard Deviation

3. High Frequency Energy Ratio

4. MFCCs 2, 7, 3, 9, 1

Applying this subset of features, rather than the full set of features, on a HNM scheme yielded the following results:



Figure 4.17 Confusion matrix for gunshot detection using feature importance with HNM for three iterations

Metric	Value
Accuracy	0.9998
Precision	0.9647
Recall	0.6833
F1-Score	0.8000

Table 4.3 Quantitative metrics for gunshot detection using feature importance with HNM for three iterations

Sequential Feature Selection

The following features were selected by backwards SFS, optimizing for FPR.

- Harmonic-to-Percussive Ratio (HNR)
- High Frequency Energy Ratio
- STD Amplitude Envelope
- MFCCs 2, 3, 5, 6, 7, 8, 12

Applying this subset of features, rather than the full set of features, on a HNM scheme yielded the following results:



Figure 4.18 Confusion matrix for gunshot detection using sequential feature selection with HNM for three iterations

Metric	Value
Accuracy	0.9997
Precision	0.9865
Recall	0.6083
F1-Score	0.7526

Table 4.4 Quantitative metrics for gunshot detection using sequential feature selection with HNM for three iterations

Bayesian Optimization

The following results and optimization curve were found using Bayesian optimization for feature selection, optimizing for FPR:

- High Frequency Energy Ratio
- STD Amplitude Envelope
- MFCCs 1, 2, 3, 7, 8, 9, 11

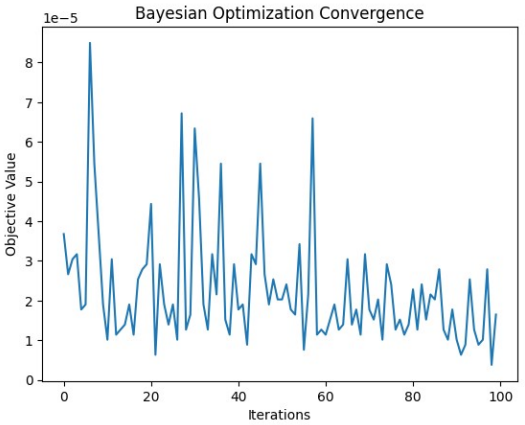


Figure 4.19 Optimization curve of Bayesian Optimization, minimizing FPR with feature selection for 100 iterations

Applying this subset of features, rather than the full set of features, on a HNM scheme yielded the following results:



Figure 4.20 Confusion matrix for gunshot detection using bayesian optimization with HNM for three iterations

Metric	Value
Accuracy	0.9997
Precision	0.9595
Recall	0.5917
F1-Score	0.7320

Table 4.5 Quantitative metrics for gunshot detection using bayesian optimization with HNM for three iterations

Hyperparameter optimization

Having examined the results for the different subsets of features, we conclude that, to minimize FPR, the following feature set (identified by backward SFS) is optimal:

- Harmonic-to-Percussive Ratio
- High Frequency Energy Ratio
- STD Amplitude Envelope
- MFCCs 2, 3, 5, 6, 7, 8, 12

Using this subset of features, we can now optimize for decision tree hyperparameters and see if further improvements are possible.
A list of the hyperparameters optimized over can be found in section 4.5.
By applying bayesian optimization over the hyperparameter space for 50 iterations, we find the following hyperparameters:

- Criterion: `entropy`
- Splitter: `best`
- Max Depth: 17
- Min Samples Split: 20
- Min Samples Leaf: 1
- Max Features: `None`



Figure 4.21 Confusion matrix for gunshot detection using sequential feature selection and hyperparameter optimization with HNM for three iterations

Metric	Value
Accuracy	0.9997
Precision	0.8947
Recall	0.7083
F1-Score	0.7907

Table 4.6 Quantitative metrics for gunshot detection using sequential feature selection and hyperparameter optimization with HNM for three iterations

4.7 Analysis

Result Analysis

We remind the reader that the primary objective of this research project is to minimize the FPR. Ideally, we aim to achieve these goals without compromising recall beyond an acceptable level. This balance is often explored through the precision-recall curve, as visualized in Figure I.16b. Here, we manually identify an acceptable optimum, prioritizing very low FPR while maintaining subjectively acceptable recall. As a result, the selected results inevitably reflect a degree of subjectivity in determining what is "acceptable."

From the perspective of minimizing FPR, the feature set found by SFS, without optimizing the hyperparameters, provides the best results.

Simply making use of feature importance, and cutting out features of low importance also performed very well. Although it has two more false positives compared to using the SFS features, it also boasts an 8% improvement in recall over using the SFS features.

Furthermore, we find that optimizing the hyperparameters of the decision tree on the training set significantly worsens the results. This is most evident when looking at the false positives, which jump from one to ten false positives when optimizing the hyperparameters. A likely explanation is that optimizing the hyperparameters on the training set causes the model to overfit and perform worse on unseen data.
All models struggle the most with sounds close to the decision boundary, such as loud banging of vehicle doors found in "Human activity", and loud thunder found in "Storm". Such analysis was made possible by listing and playing the sound files misclassified as false positives by the model.
Since the model focuses on reducing false positives, there is almost paradoxically more data to analyze when looking at the false negatives of the model.
The total positive data collected can be found in table 4.1. Unfortunately, since the samples are randomized when placed in the training, validation, and test splits, no samples from 350 meters are found in the test set. The 120 positive samples found in the test set are presented in the following table:

Weapon Type	**10m**	**50m**	**150m**
AR (Assault Rifle)	13	9	13
Shotgun	13	12	10
Hunting Rifle	6	6	2
Total	32	27	25

Table 4.7 Number of Positive Samples in the test set by Weapon Type and Distance

The best model had 47 false negatives on the test set:

Weapon Type	**10m**	**50m**	**150m**
AR (Assault Rifle)	9	3	4
Shotgun	6	5	3
Hunting Rifle	3	0	1
Total	18	8	8

Table 4.8 Number of False Negatives by Weapon Type and Distance

To analyze trends, we can define the false negative rate per category as:

$$\text{False Negative Rate} = \frac{\text{Number of False Negatives}}{\text{Number of Positive Samples}} \cdot 100\%$$

The model struggled the most with assault rifles at 10 meters, misclassifying about 69% of them. Shotguns also showed high false negative rates at 10m and 50m (46.2% and 41.7%, respectively). Although the model performed better with hunting rifle samples at 50 meters, higher misclassification rates were observed at 10m

Weapon Type	10m	50m	150m
AR (Assault Rifle)	69.2%	33.3%	30.8%
Shotgun	46.2%	41.7%	30.0%
Hunting Rifle	50.0%	0.0%	50.0%

Table 4.9 False Negative Rates by Weapon Type and Distance

and 150m. Overall, there was a general decrease in the false negative rate as distance increased, apart from hunting rifles, which had limited data at longer distances.
The analysis indicates that the model's performance varies with weapon type and distance. The higher false negative rates at shorter distances—particularly for assault rifles and shotguns at 10m—indicate that the model struggles with gunshot characteristics recorded close to the firing site. Paradoxically, clearer recordings at close range can yield more challenging samples; for instance, those manually labeled at the start or end of a gunshot may only contain subtle traces of the sound, making detection more difficult. In terms of weapon type, assault rifles showed the highest overall false negative rates, suggesting that the model may be less effective at detecting AR gunshots compared to shotguns and hunting rifles. Meanwhile, the inconsistent performance observed for hunting rifle samples, especially at 150m, could be attributed to the smaller number of samples available for both training and testing.

Feature Analysis

An important consideration in feature selection is the computational cost associated with feature extraction. Mel-Frequency Cepstral Coefficients (MFCCs), while effective, are computationally intensive features. If alternative methods to decrease the FPR are employed—for example, the method discussed in Section 4.7—it may be feasible to use less computationally expensive features.
MFCCs are widely used in audio processing because they effectively capture the timbral aspects of sounds, which are crucial for tasks such as speech recognition. However, the extraction of MFCCs involves several computationally intensive steps:

1. Framing and Windowing: The audio signal is divided into overlapping frames, and each frame is multiplied by a window function (e.g., Hamming window) to reduce spectral leakage.

2. Fast Fourier Transform (FFT): Each windowed frame undergoes an FFT to convert the time-domain signal into the frequency domain.

3. Power Spectrum Calculation: The magnitude squared of the FFT output is computed to obtain the power spectrum.

4. Mel Filter Bank Processing: The power spectrum is passed through a set of filters spaced according to the Mel scale, which approximates the human ear's critical bandwidths.

5. Logarithm of Mel Energies: The logarithm of the filtered energies is taken to compress the dynamic range.

6. Discrete Cosine Transform (DCT): A DCT is applied to decorrelate the log Mel energies, resulting in the MFCCs.

Each of these steps requires significant computational resources, with the FFT and DCT being particularly intensive. This computational burden is especially problematic when real-time processing is necessary. Moreover, even if only a subset of the MFCC coefficients is needed for analysis, the entire set must still be computed. This is because the MFCCs are derived from the full DCT of the logarithm of the Mel-filtered energies, making it impossible to calculate individual coefficients in isolation.
To reduce computational load, replacing MFCCs with features that are less demanding to compute but still capture essential characteristics of the audio signal is an option. Potential alternatives include:

- Zero-Crossing Rate (ZCR): Measures the rate at which the signal changes sign, useful for detecting noisy or percussive sounds. Computed directly from the time-domain signal, making it computationally inexpensive.

- Root Mean Square (RMS) Energy: Represents the signal's average power, providing a simple measure of loudness.

- Temporal Features: Such as mean, variance, skewness, and kurtosis of the amplitude envelope, which can be calculated directly from the time-domain signal.

- Entropy of Energy: Measures the randomness of the energy distribution in the signal.

By focusing on features that can be computed directly from the time-domain signal or require minimal spectral analysis, the computational requirements of the feature extraction process can be significantly reduced. Preliminary results showed that doing so indeed greatly reduces computational cost but comes at the expense of a slight increase in false positives (up to ten).
Many of the papers and research studies in the field of audio signal processing and gunshot detection (such as those discussed in Section 4.2) do not prioritize low-compute features because they are often developed in well-funded areas such as the military or large technology companies, where computational resources are generally not an issue. These systems can afford to utilize complex algorithms and

high-dimensional feature sets without significant concern for processing time or energy consumption.
In contrast, this work focuses on developing a system for use on the savannah, catering to non-profits and conservation efforts where resources are limited. This sets entirely new requirements for building cost-effective and low-compute products. Devices deployed in such environments need to be energy-efficient—most likely solar-powered—and capable of processing data in real-time or near real-time without relying on extensive computational infrastructure.
The unique challenges of the savannah's acoustic environment—including diverse animal sounds, weather conditions, and human activities—highlight the importance of experimenting with real-world data specific to the use case. While academic papers and existing research offer valuable insights and starting points, they cannot substitute the necessity of working with data that reflects the complexities of the actual environment. As far as the authors know, such datasets are neither represented in other studies nor openly available.

Estimating Deployment on System-on-Chip

Estimating the inference time on a microprocessor is crucial for deploying the developed system on a system-on-chip (SoC) in the savannah. A SoC integrates a processor, memory, and other essential components into a single chip, enabling compact, energy-efficient devices for real-time environmental monitoring. This ensures the system can operate effectively within the hardware constraints, particularly in resource-limited environments where processing power and energy consumption are critical.
The Sony Spresense microcontroller board was chosen for this analysis due to its application in similar contexts [4]. Its popularity makes it an appropriate benchmark for estimating system performance on comparable hardware.
To estimate the inference time on the target microprocessor, execution time is profiled on the development hardware and then scaled according to the Sony Spresense's specifications. This scaling accounts for differences in processor speed, architecture, and computational resources, providing an approximate inference time for deployment on the SoC.
This estimation helps determine whether the system meets real-time processing requirements and identifies potential bottlenecks before deployment, ensuring the system will function as intended in the target environment.
The laptop used for development has the following specifications (see appendix K for more details):

- CPU Model: 11th Gen Intel Core i5-1135G7 @ 2.40 GHz
- CPU Max Frequency: 4,200 MHz

We note that no manual optimization was performed on the laptop hardware. The execution times on this hardware:

- Feature extraction time: ≈ 123.8 ms
- Inference time: ≈ 0.3 ms

The Sony Spresense has the following specifications:

- CPU Model: ARM Cortex-M4F
- CPU Frequency: 156 MHz

To estimate execution time on the Sony Spresense, we scale the measured times by the ratio of the laptop's CPU frequency to that of the Spresense:

$$\text{Scaling Factor} = \frac{\text{Laptop CPU Freq}}{\text{Spresense CPU Freq}} = \frac{4,200 \text{ MHz}}{156 \text{ MHz}} \approx 26.92$$

The estimated feature extraction time on the Spresense:

$$\begin{aligned} \text{Feat. Extract. Time}_{\text{Spresense}} &= \text{Feat. Extract. Time}_{\text{Laptop}} \cdot \text{Scaling Factor} \\ &= 123.8 \text{ ms} \cdot 26.92 \\ &\approx 3.3 \text{ seconds} \end{aligned}$$

The estimated inference time per sample on the Spresense:

$$\begin{aligned} \text{Infer. Time per Sample}_{\text{Spresense}} &= \text{Infer. Time per Sample}_{\text{Laptop}} \cdot \text{Scaling Factor} \\ &= 0.3 \text{ ms} \cdot 26.92 \\ &\approx 8 \text{ ms} \end{aligned}$$

In total, the estimated total processing time on the Spresense is approximately 3–4 seconds.

Transitioning from a confusion matrix to the real world

When designing systems for real-world applications, bridging the gap between theory and practice is crucial. For instance, a ranger may find it challenging to interpret the implications of a confusion matrix in practical terms. Thus, it becomes essential to understand how theoretical results translate into real-world impact.
To facilitate this understanding, a simplified script was created. The script models the Makulu Wildlife Sanctuary, which spans an area of 3 km by 10 km. It places 100 microphones evenly across the area and uses the optimal confusion matrix values identified in the results section.

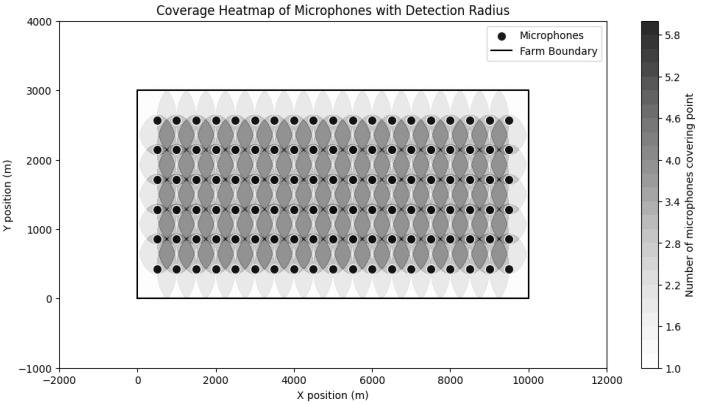


Figure 4.22 Plotted simulation map

Understanding the practical implications of model performance is critical for real-world applications. Key metrics such as the expected time between false positives, the probability that a real gunshot is detected, and the probability that a detected event is truly a gunshot provide meaningful insights into system reliability and usability.

The expected time between false positives $\mathbb{E}[\text{FP}]$ is particularly important for assessing how often a ranger or monitoring system might needlessly respond to false alarms. This metric is calculated as the inverse of the product of the daily empty sample rate (S_{empty}) and the false positive rate (FPR):

$$\mathbb{E}[\text{FP}] = \frac{1}{S_{\text{empty}} \cdot \text{FPR}}$$

Given that a new sample is recorded every second, the daily empty sample rate is calculated as follows:

$$S_{\text{empty}} = 100\,\text{microphones} \cdot 1\,\text{Hz} \cdot 24\,\text{hours/day} \cdot 3600\,\text{seconds/hour} = 8,640,000\,\text{samples/day}$$

The false positive rate is derived from the confusion matrix as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = \frac{1}{1 + 171,185} \approx 5.84 \cdot 10^{-6}$$

Substituting these values into the equation for $\mathbb{E}[\text{FP}]$:

$$\mathbb{E}[\text{FP}] = \frac{1}{8,640,000 \cdot 5.84 \cdot 10^{-6}} \approx 0.02\,\text{days}$$

This calculation indicates that, even with only one false positive every 171,186 samples, an array of 100 microphones would generate a false alarm approximately thirty minutes.

For illustrative purposes, a detection radius of 500 meters is assumed. The probability that a shot is detected by at least one microphone, denoted as the system true positive rate TPR_{sys}, is calculated based on the individual true positive rate of each microphone, TPR, and the number of microphones within range that detect the shot, M_{detect}.
For each simulated shot, M_{detect} is determined by identifying the microphones located within the shot's detection radius. The probability that at least one of these microphones detects the shot is given by:

$$\text{TPR}_{\text{shot}} = 1 - (1 - \text{TPR})^{M_{\text{detect}}}$$

In this formula, $(1 - \text{TPR})^{M_{\text{detect}}}$ represents the probability that none of the M_{detect} microphones detect the shot. Averaging TPR_{shot} over 1 million simulated shots yields an estimated system true positive rate of $\text{TPR}_{\text{sys}} \approx 0.9990$. This high value highlights the effectiveness of the microphone array in detecting gunshots within the assumed environment.

The precision, $Precision_{\text{shot}}$, or the probability that a detected event is actually a shot, is calculated as the ratio of the expected true positives per day, E(TP), to the total detections per day, which includes both true positives and false positives. This calculation assumes approximately one true gunshot occurs every 28 days, based on empirical observations at the site.

$$\text{Precision}_{\text{shot}} = \frac{\mathbb{E}[\text{TP}]}{\mathbb{E}[\text{TP}] + \mathbb{E}[\text{FP}]}$$

Substituting values:

$$\mathbb{E}[\text{TP}] = \text{TPR}_{\text{sys}} \cdot \frac{1}{\mathbb{E}[\text{days between shots}]} = 0.9990 \cdot \frac{1}{28} \approx 0.0357 \,\text{shots/day}$$

$$\mathbb{E}[\text{FP}] = S_{\text{empty}} \cdot \text{FPR} = 8,640,000 \cdot 5.84 \cdot 10^{-6} \approx 50.5 \,\text{false positives/day}$$

$$\text{Precision}_{\text{shot}} = \frac{0.0357}{0.0357 + 50.5} \approx 0.0007$$

These calculations provide the expected time between false positives as 0.02 days, the probability that a shot is detected as 0.9990, and the probability that a detection is actually a shot as 0.0007.

Meta-Models

Multi-microphone detection. To address the high false positive rate in the system, an effective strategy involves implementing a meta-model that requires multiple microphones to confirm a detection before generating an alert. Specifically, requiring

at least N microphones to detect a shot within a short time window Δt can significantly reduce the number of false positives while maintaining a high true positive rate.
A true gunshot produces a sound that propagates through the environment and can be detected by multiple microphones within a certain radius. In contrast, false positives, such as misclassified environmental noises, are typically random and uncorrelated events occurring independently across microphones, with some exceptions. For example, an elephant snapping a branch could potentially be heard by several microphones as it is a loud sound.
This difference can be leveraged by designing a system where an alert is raised only if multiple microphones detect an event simultaneously. Such an approach reduces the likelihood of false positives, as the probability of multiple independent false detections occurring at the same time is low.
Let:

- M be the total number of microphones.
- TPR be the true positive rate of a single microphone.
- FPR be the false positive rate of a single microphone.
- N be the minimum number of microphones required to confirm a detection.
- Δt be the time window within which detections must occur to be considered simultaneous.

For a true gunshot, the probability that at least N out of M_{detect} microphones within detection range detect the shot is very high. Given that TPR ≈ 0.9990 and assuming $M_{\text{detect}} \geq N$, the value of $P_{\text{TP}}(N)$ remains close to 1. This ensures a high probability of detecting actual gunshots.

For false positives, since these events are assumed independent across microphones, the probability that at least N microphones produce a false positive within the time window Δt decreases exponentially with N:

$$P_{\text{FP}}(N) = \binom{M}{N} (\text{FPR} \cdot \Delta t)^N$$

With a small FPR and increasing N, $P_{\text{FP}}(N)$ becomes negligibly small, significantly reducing false positives.

For illustrative purposes, consider $N = 2$ and $\Delta t = 1$ second. Given a false positive rate FPR $= 5.84 \cdot 10^{-6}$:

$$P_{\text{FP}}(2) = \binom{M}{2} \left(5.84 \cdot 10^{-6}\right)^2$$

For $M = 100$, this results in approximately 0.0146 false positives per day—a significant reduction from the initial rate of 50.
Assuming one true gunshot every 28 days:

$$\mathbb{E}[\text{TP}] = \frac{1}{\mathbb{E}[\text{days between shots}]} = \frac{1}{28} \approx 0.0357 \,\text{true positives/day}$$

The updated precision can be expressed as:

$$\text{Precision}_{\text{shot}} = \frac{\mathbb{E}[\text{TP}]}{\mathbb{E}[\text{TP}] + \mathbb{E}[\text{FP}_{\text{2 sensors}}]}$$

Substituting the values:

$$\text{Precision}_{\text{shot}} = \frac{0.0357}{0.0357 + 0.0146} \approx 0.709$$

This represents a significant improvement over the initial precision.

Optimizing Detection Radius and Microphone Placement. To maximize the number of microphones within detection range (M_{detect}):

- Enhance microphone sensitivity or use higher-quality equipment to detect sounds over greater distances.
- Place microphones on high altitude spots such as *koppies*. These elevated locations can increase the detection range due to reduced obstructions and better line-of-sight, allowing microphones to pick up sounds from further distances.

Adjusting the Time Window Δt. A slightly larger Δt (e.g., 2–3 seconds) can accommodate variations in detection times due to signal propagation delays and processing times, ensuring that simultaneous detections are captured. Furthermore, having a larger time-scale on which to perform classification, the models can in theory learn and capture more complex edge cases, such as elephants breaking sticks. For a refresher on such edge cases, see section F.1.

Incorporating Human Verification. To further enhance the reliability of the detection system, incorporating a layer of human verification can be highly effective. Once positive samples are confirmed by the required number of microphones within the time window Δt, the audio clips can be sent to a ranger for additional confirmation. This is also requested by rangers on site, as seen in section F.1
This human-in-the-loop approach offers several benefits. Rangers can listen to the audio clips to confirm whether they are indeed gunshots or false positives caused by

environmental noises or wildlife activities. Feedback from rangers on ambiguous sounds can help refine the model by incorporating new edge cases into the training data. Finally, involving rangers in the verification process increases their trust in the system, as they have direct input before any action is taken.
By integrating human verification, a hybrid system is created that combines the efficiency of automated detection with the discernment of human judgment. This approach can hopefully reduce false positives and ensure that alerts are both accurate and actionable.
For transitioning to real-world implementation of a microphone-based gunshot detection system, the following recommendations are proposed:

- Implement Multi-Microphone Confirmation: Require at least $N = 2$ detections within $\Delta t = 2$–3 seconds to trigger an alert.
- Monitor System Performance: Regularly assess true positive and false positive rates to adjust N and Δt as needed.
- Balance Detection and Responsiveness: Identify an optimal trade-off between reducing false positives and maintaining a high true positive rate.
- Incorporate Human Verification: Include a step where rangers review and confirm detections to enhance accuracy and system trust.

5

Case 2: Elephant detection

5.1 Background

This case is about using geophones - ground vibration sensors - to detect footsteps from animals and humans. In this work, only elephant footsteps have been considered. The applications of such a device could be to track animals, detect poachers walking on the property, and more, as explored in Subsection 5.3. The system has the potential to be more accurate and have a longer detection range than cameras in bushy areas, as the vegetation blocks the view for the camera.

Geophones are devices for measuring seismic vibrations. They consist of a magnet surrounded by a coil. Outside of the coil is a housing, to which the magnet is attached. The coil is suspended on springs. When the ground moves, the case and magnet move with it, while the coil stays still. The relative movement induces a voltage over the coil proportional to the velocity - so the signal amplitude is approximately proportional to ground velocity. Geophones provide accurate and sensitive measurements of vibrations from their natural frequency, around 10 Hz, to the spurious frequency, around 250 Hz [25].

5.2 Literature Review

Earlier master theses on this topic

Two other master theses have been written on the topic of elephant detection, in relation to Smart Savannahs. They both use the same type of geophones and ADCs as this work, see Section 5.6. The setup used in this work is the same as that used by Goderik and Westlund [5].

Sjövik and Wahldehov [26] made measurements on Asian elephants in Kolmården, Sweden. They note a wave propagation speed of 165.0 m/s in dry ground in the area of Vallamassivet, Sweden. This was found as the average over 12 samples. *We note that the standard error is 1.8 m/s*. Furthermore, they report that the signal from a

footstep has a large amplitude for around 350 ms. They also find that the elephant footstep has the highest spectral density in the range 10-30 Hz. All of their footsteps were recorded within 35 m of the geophone. Their training dataset consists of 18 elephant footsteps and 11 human footsteps, while their validation dataset consists of 17 elephant footsteps and 11 human footsteps. These samples were extracted as 700 ms long segments overlapping each other by 350 ms. They note that although their method is 100% accurate on the validation samples, their method misclassifies one of the positive training samples, and question whether the method would be accurate on footsteps more than 35 m away.

Goderik and Westlund [5] made measurements on Asian elephants in Kolmården, as well as on African elephants in Kenya. They note a wave propagation speed of 161.7 m/s in dry ground in Kolmården, Sweden. This was found as the average over 10 samples. *We note that the standard error is 3.8 m/s*. They examine the frequency contents of footsteps from different animals, and find high energy in the range of 5-30 Hz for elephants, 5-40 or 10-70 Hz for humans, and 5-100 Hz for rhinos. Water buffalos have three distinct peaks around 50, 100, and 150 Hz. Kulans and camels have the highest energy contents for frequencies over 100 Hz, with still high energies at their Nyquist frequency of 237. The detection performance is evaluated on footsteps from Kolmården. All of their elephant footsteps were recorded in the range of 5-35 m away from the geophone. Their training dataset contains 109 elephant footsteps, while their validation dataset contains 57 footsteps from each of elephant, human, rhino, and camel, and 49 footsteps from kulan. Samples were extracted from continuous signals based on kurtosis thresholds, so their sample counts are of segments that met this threshold, and do not include rejected segments in between. When the threshold was met, a sample of length 270 ms was extracted. The validation elephant footsteps were recorded by other students a year before, possibly the other thesis. They achieve an accuracy of 89% in detecting elephant footsteps.

Seismic properties of Asian elephant (Elephas maximus) vocalizations and locomotion

O'Connel-Rodwell et al. [27] note that the amplitude of Rayleigh waves disperses as $1/\sqrt{r}$, and that a foot stomp from an elephant traveled with 264 m/s as a seismic Rayleigh wave. They draw this conclusion, from the observation that only Rayleigh waves could propagate this slowly. This speed is much higher than those measured in the previous theses.

The locomotor kinematics of Asian and African elephants: changes with speed and size

Hutchinson et al. [28] find that the stride length (how long the foot contacts the ground) of African elephants varies between 0.26 and 4.00 seconds, and the stride frequency (whole gait cycles) between 0.21 and 1.90 Hz.

Using seismic sensors to detect elephants and other large mammals: a potential census technique

Wood et al. [29] record geophone signals from elephant footsteps, and note that the front legs produce higher amplitude signals than the hind legs.

Seismic savanna: machine learning for classifying wildlife and behaviours using ground-based vibration field recordings

In the study Seismic savanna [30], Szenizer et al. use seismometers, devices that are similar to geophones, to detect several animals. In their study, they use a sample rate of 200 Hz, and place the seismometers 70 cm deep into the ground. They use several models, and find that the most accurate is a SqueezeNet [31] pretrained on the ImageNet database and fine-tuned on spectrograms, with a single fully connected classifier layer. To evaluate their models, they use balanced accuracy (BA), which is more descriptive than accuracy for imbalanced datasets. For detecting elephants, they achieve close to 90% BA up to 80 m distance, which then decreases to 82% at 100 m, and decreases further to 73% for 150 m. They evaluate data augmentation techniques that they deem appropriate for ground vibration data; source superposition (addition of samples from the same class), and addition of noise from empty recordings to samples with a class (the result should still belong to the same non-empty class), but do not find improvements with these. A crucial consideration they point out is that their data has been labeled using camera traps covering small parts of the detection areas around the geophones, and that this may lead to label corruption.

Relevance of papers

The earlier master theses [26] [5] were used as guidance for our work, and their measured wave propagation speeds have been used directly. Their measured event lengths and spectral densities, and their prediction methods have been considered. The dispersal and wave propagation values from O'Connel-Rodwell [27] were noted. The measurements on African elephant strides [28] are relevant for the signal they produce, and what cadences to expect. Wood et al. [29] taught us to expect larger signals from the elephants front legs, with smaller from the hind legs in between. Seismic savanna gave information on possible sensing distance and data augmentation methods. They also found a model that was very successful for classification of footsteps, providing inspiration for further work based on this thesis [30].

5.3 Interviews with rangers

The interviews serve to give perspective on and motivate the use case, as well as highlight potential problems. The interview findings for this case are detailed more extensively in Appendix F.2. The observations in this section are all from the interviewees. None of the interviewees has worked with footstep detection systems before.

Several use cases are uncovered. One is poacher detection. Different interviewees suggest different placements; either close to the property border, or where animals are known to walk. One interviewee mentions that the circular detection radius may make detection along the fence line suboptimal. Special fiber optical fence line systems are known to exist, but also to be prohibitively expensive. It is mentioned that poachers use nonnormal footwear (such as diapers), and walk in animal tracks, to hide their own tracks. For this use case, it comes up that estimating pace and direction of travel is desirable. It is also mentioned that poachers tend to initially run a distance of 100-200 m away from the crime scene, so detecting running humans is interesting, maybe even as a separate category from walking humans. It is mentioned that some poachers bring dogs. Another proposed use case of a footstep detection system is as a new source of animal tracking information. This could be useful, as wearable trackers are often unreliable, and not worn by all animals. Elephants, rhinos, and buffalos are mentioned as interesting species to detect. This could help in figuring out their favorite spots and whereabouts, which could make the placement of other sensors and security resources more informed. The desired placement for this use case would be along known walking paths, near watering holes, feeding spots, and mud wallows. A third use case is to detect when an elephant is getting close to a valuable tree. This would then trigger a defense system. The interviews highlight that it is vital to differentiate between the different footsteps, and that communication with other sensors could be useful.

For animal detection, it is noted that there may be other signatures than footsteps. Such a signature could be, for example, an elephant rumble, or a kick to the ground from a rhino. It may also be interesting to detect these special signatures, as they can signal mating rituals, or hole digging, which would be of interest to know about.

Several potential problems come up. One is false alarms from other animals, humans that belong in the sanctuary, and vehicles. Another is that elephants detect human smells on electronics and attack them, *on this note we refer to Figure 5.4*. Beyond elephants, the device also needs to be resistant to water and insects. One interviewee notes that soil erosion should be considered when choosing the geophone placement.

5.4 Problem definition

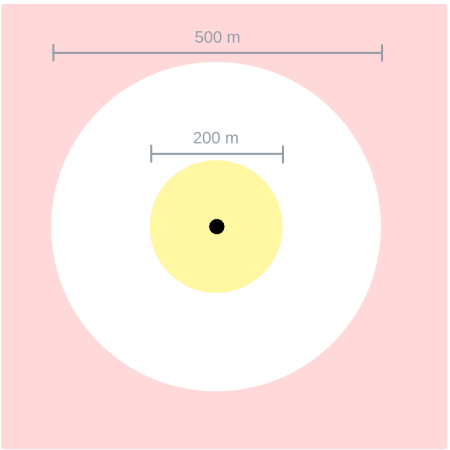


Figure 5.1 Illustration of areas defining positive and negative samples. Steps within the yellow area define positive samples. If all elephants are in the red area, the sample is negative. Samples that do not fit these categories are excluded.

The problem our model should solve can be defined in several ways, with different pros and cons. The definition in this work was chosen to strive for a model that is accurate in a given distance range, within which its performance can be precisely evaluated, but that is not necessarily limited to this range. The definition of positive and negative samples is illustrated in Figure 5.1. We define the set of positive samples S_p as

$$S_p = \{s : s \text{ contains an elephant step taken in } A_{\text{yellow}}\}. \tag{5.1}$$

The precise formulation of "contains an elephant step", is that the sample contains the complete signal measured 0-350 ms after the seismic wave from a step arrives to the geophone. So if the elephant takes a step at time t, and it takes Δt for the signal to arrive to the geophone, then a sample containing the time segment $[t+\Delta t, t+\Delta t+350]$ would be positive.The negative set S_n is defined as

$$S_n = \{s : \text{all elephants are in } A_{\text{red}} \text{ during } s\}. \tag{5.2}$$

It should be noted that the positive and negative samples do not cover the full space of possible samples. We have defined a third category, which is referred to as "half-negative", and contains all the samples that do not fit as either positive or negative. The reason for introducing this category is as follows. The inner ring, here defined with a radius of 100 m, is set to a distance within which we expect/wish to be able

to detect elephant footsteps. The outer ring is set to a distance at which we expect to not be able to detect footsteps, and in fact, do not expect the footstep signal to show up in the measurement at all. Thus, we can be sure that negatives in the positive samples are indeed false negatives, and positives among the negatives are indeed false positives. The reason for counting samples within the inner ring when the elephant is not walking as half-negatives, is that there may be steps that we cannot see; sometimes the elephants take small steps to adjust the direction in which they face. It is important to avoid these positives mislabeled as negatives, as if the model fits to them, we may get false negatives in prediction for similar samples. If we only used the inner ring, we would have a problem that was not very usefully defined - is a "false positive" when the elephant is 101 meters away really an error we want to suppress? Moving the inner ring to the outer could have benefits, which are discussed in Section 5.10, however, this also introduces complications. Introducing some sort of radius that defines positives is a necessity; consider a scenario where we can see very far, but can only detect footsteps 100 m away with the geophone. Then, a footstep seen 10 km away from the geophone should obviously not generate a positive sample, as it will not show up in the signal and we cannot expect to detect it. Since the elephants are always *somewhere* and very likely at least one of them is moving, this is not a useful problem formulation. The sizes of the radii have been chosen based on three factors: information from Seismic Savanna [30], the earlier theses [26] [5], visual inspection of our own data, and how far away we were able to supervise the data collection. We have made a relatively optimistic judgment of how far a good model should be able to detect the elephant footsteps. In this thesis, we had to approximate the outer radius due to the bush limiting how far away there was visibility. Instead, we defined the negative samples as 10 minutes before the elephant came into view and 10 minutes after it went out of view. This relates to an observation we made, that we here take as an approximation; that elephants mostly traverse open areas and roads. It should be noted that the reliability of these approximations is increased by the fact that all supervised setups had a watering hole in view of the camera. Thus, it is likely that nearby animals would at some point pass the watering hole; thus coming into view. So if no animal is seen, it likely was not nearby either. We have not used the half-negatives at all, however, they could be used to make qualitative assessments of how far away from the geophone footsteps could be detected, and find out if the inner ring could be pushed further away.

5.5 Hardware concept

The hardware concept consists of two base components; a unit that reads data from a geophone, and a camera that supervises the area where the geophone is placed. One of each unit, together with a solar panel for power supply, are shown in Figure 5.2a. A geophone element and cable is shown in Figure 5.2b. The geophone unit samples at 1000 Hz. The camera takes one photo per second.



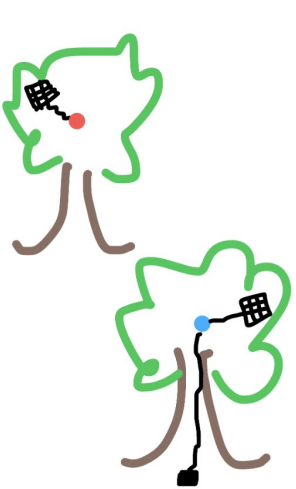
(a) A solar panel, a camera, and a unit that collects data from a geophone.

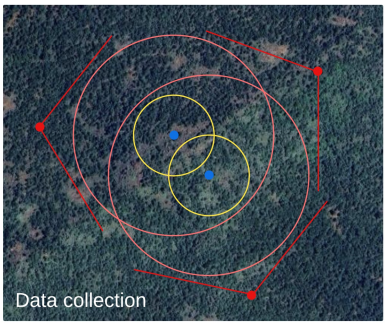


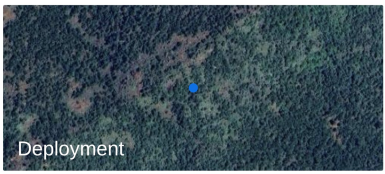
(b) The geophone element and its cable, terminating in a 3.5 mm connector.

Figure 5.2 The hardware for Case 2.

The conceptualized hardware setup is shown in Figure 5.3. As shown, both the units that read the geophones and the cameras are powered by solar panels, together with which they are placed high up in trees. In Makulu, the trees are distributed quite sparsely, more about this can be read in Section 3.3. The reasons for this placement are: To have the gear out of reach from animals, to facilitate good network connectivity, and to enable placement of the solar panel where it receives as much sunlight as possible. The conceptual geographic placement is seen in Figure 5.3b. It is devised to facilitate solving the problem according to the problem definition, by making it possible to accurately determine when the elephant is within the inner and outer rings. Preferably, at least two cameras should be used, so that the location of the elephant (and thus the distance to the sensor) can be determined accurately. In our work, only one camera has been available, so instead Google Maps has been used to estimate the distance using landmarks, such as trees. In a real-use scenario where simply detecting an elephant is the aim, only a single geophone unit would be placed in an area. If degree-of-arrival (DoA) estimates are interesting, as in the earlier theses on this topic [26] [5], three geophone units would be placed. With enough space between the geophone, localization to a point would be possible.







(a) Depiction of how the hardware should be placed in trees. The actual geophone element is drawn as a black box. The checkered rectangles are the solar panels.

(b) Depiction of how the hardware should be placed geographically. The multiple cameras and geophones demonstrate the modularity of the concept. The circles are the same as in 5.1. ©2025 Bilder, ©2025 Airbus, Maxar Technologies.

Figure 5.3 The concept for the geophones. Red dots represent cameras (red lines the edges of their fields of view), and blue dots geophone units.

5.6 Hardware

Geophone unit

The geophone unit is seen in Figure 5.2a, and consists of the following parts:

- SM-24 geophone element
- ADS1256; a 24-bit, low noise, analog to digital converter (ADC)
- LilyGo T-SIM7000G unit, based on the ESP32 microcontroller
- Sandisk Ultra 64GB microSD-card
- Battery
- Watertight plastic container

The geophone element provides an analog signal, that is read by the ADC. The element is soldered to a 5 m long coaxial cable ending in a two-pole 3.5 mm headphone connector. A 3.5 mm headphone jack was soldered to the ADC using cables. The 3.5 mm jack was glued on the outside of the container lid, aligned with the container wall, to avoid having the connector sticking out, and therefore vulnerable to forces from the side. The cables were drawn out to the jack through holes made in the container lid, after which the holes were sealed, making the setup waterproof. The LilyGo unit and the battery were both glued to the inside of the container lid. The setup, excluding the container, was inherited from the thesis by Goderik and Westlund [5], when it was found that the ADC in the unit we made for our thesis was insufficiently sensitive to small signals. The programmable gain amplifier (PGA) of the ADC was set to 64, the maximum value, to give the highest possible resolution of the small geophone signal. At this setting, the voltage resolution was 4.66 μV per bit. Note that some hardware modifications were performed throughout the project due to breakages.

According to the datasheet of the geophone used in our work, the sensitivity drops quickly below 10 Hz [32]. It also says "Extended spurious over 240 Hz, allowing full bandwidth at 2-ms sampling", indicating that there may be spurious signals present with vibrations at frequencies above 240 Hz. Thus, measuring faster than 500 Hz might not add any information, as strong signals of a higher frequency than 240 Hz would also give a lot of spurious signals.

The ADS1256 is a 24-bit analog-digital converter with up to 23 bits of noise-free resolution. The converter is a 4th-order delta-sigma modulator, followed by a programmable digital filter. With the programmable gain amplifier (PGA) set to 64 times amplification, there is an offset drift with a magnitude of between 0 and 14 nV/°C temperature offset from 25°C. The gain error is on the scale of 10 ppm. With the analog input buffer off, PGA at 64, and a sample rate of 1 kHz, the noise has a normal distribution with a standard deviation of around 20 bits. At a PGA setting of 64 and with the analog input buffer on, the input-referred noise is 0.367 μV root-mean-square (RMS) at 500 Hz, and 0.486 μV RMS at 1000 Hz, as an effect of reduced averaging. *We note that* $0.486/\sqrt{2} = 0.344$*, so sampling at 1000 Hz and downsampling to 500 Hz by averaging samples will produce approximately the same signal, as sampling at 500 Hz, assuming that the noise is not very autocorrelated*. The frequency response is 1 (normalized) at 0 Hz, has zeroes at multiples of the sampling frequency, and peaks approximately in the middle between those zeros. If all the signal content is below the sampling frequency, only the frequency response from 0 to the sampling frequency needs to be studied. At 1000 Hz, the -3dB (half the maximum response) bandwidth is 441 Hz (which is quite close to the Nyquist frequency of 500 Hz).

Camera unit

A camera unit is shown in Figure 5.2a, and consists of the following parts:

1. LilyGo T-SIMCAM unit, based on the ESP32-S3 microcontroller
2. OV2640 camera with 120° IR-CUT or 200° lens
3. Sandisk Ultra 32GB microSD-card
4. Voltage regulator
5. Hunter Solar Panel Plus SP150
6. Plastic container

The LilyGo was powered by a solar panel with a built-in battery, through a voltage regulator. The camera modules could be exchanged easily, as they were attached to the LilyGo with a snap-on digital video port (DVP) interface. The camera units were inherited from another project within Smart Savannahs, and fetched from different locations within the sanctuary, where they had been installed.

Hardware problems and fixes

Some problems were encountered with the hardware, most importantly the geophone unit was destroyed by a group of elephants when the setup was finalized, putting an end to the data collection. This is shown in Figure 5.4, which is a crop of a frame from the supervising camera. Another major issue was that the measurements had a lot of noise or glitches in them, the cause of which was not identified. The noise was seen also when no input signal was provided, indicating that the issue was with the ADC and not the geophone.

Some problems were encountered also with the cameras, including the units breaking (seemingly due to overheating), and the camera driver crashing on detailed scenes with low values of JPEG compression. A more detailed description of all problems is available in Appendix C.

Software for data collection

The software was written in Arduino C, and structured so the same code package could be used for both the geophone and camera units, with preprocessor directives separating the synchronization roles. The functionality of the code is visualized and explained more in Appendix A. The sample rates and data structure dimensions are set in each sensor header file, so adding more sensors and functionalities in the future is easy to do with the same code package.



Figure 5.4 The elephants destroying the geophone unit. The unit is in the red circle.

5.7 Data

Data was collected on 20 occasions, spread over 13 days. In total, around 81 hours of geophone data were collected, of which 62 hours and 20 minutes have supervision and can be used. There are 20 minutes of data where an elephant is present, from one single occasion. In no case did the field of view of the camera cover the whole inner circle, due to the visibility restriction from bushes and trees; motivating the approximations described in Section 5.4. The first two occasions were collected inside the lodge area and were manually supervised, as no camera was available. The next 16 occasions were collected on the border of the lodge area, where the border faced an open area with a watering hole. Here, the supervised range was increased by utilizing the fenced-in lodge area to ensure that no elephant would appear behind the camera.Within this fenced-in area, there were only humans, nyalas, and a dog (under normal conditions). This, however, introduced the issue of footsteps from these other species occurring, without them being seen, making manual verification of the labels very difficult. The last two occasions were collected about 3.5 km away from the lodge area, next to a road on the Makulu property that was often traversed by animals. The geophone was placed close to a watering hole. Here, a square with a side of around 200 m had been cleared in the bush, giving good visibility. This is the only area we found within Makulu, where viewing the whole outer ring is close to possible. To properly supervise this entire area, however, more than one camera would be required. With one camera, the made approximations and assumptions are necessary. An exhaustive list of every dataset and illustrations of where they were collected are found in Appendix D.

5.8 Modeling pipeline

The modeling has been performed aiming to maximize the F1-score. The choice of F1-score as the measure to focus on is based on the intended use case. Intuitively, precision is the most interesting measure. For poacher detection, it is reasonable to want to maximize the probability that a warning, caused by a detection, is in fact a threat. Also for the elephant (or other animal) detection, which we can imagine would present dots on a map, it is reasonable to look for as trustworthy dots as possible. However, if the detector gives one true positive out of 10000, it is not very useful, even if it has a perfect precision score. The F1 score takes into account both precision and recall, so it is a good measure for our modeling. Other F_β scores could be considered for further work, for example with lower weight on recall to emphasize precision.

Preparation of data

The preparation of the data consists of the following steps:

1. Denoise the geophone data
2. Synchronize the camera and geophone data
3. Label the data.

Denoising. Figure 5.5 shows the noise in a segment of data. The signal was attempted to be restored using the following algorithm:

1. Identify periodic glitches and remove them.
2. Identify regions of samples with offset means and remove them.
3. Reset the drifting mean.
4. Restore the samples with offset mean.
5. Find and interpolate single-sample glitches. These are found:
    - near the periodic glitches
    - as artifacts at the edges of the restored mean glitches
    - independently
6. Linearly interpolate the periodic glitches.

The parameters have been set such that the restoration is restrictive, to avoid removing real signals. The details of the algorithm are intricate and have been left out of the report, as the method is very specific to the faulty hardware used in our thesis. The results after each of the steps (except for after restoring the offset mean samples) are exemplified in Figure 5.9.

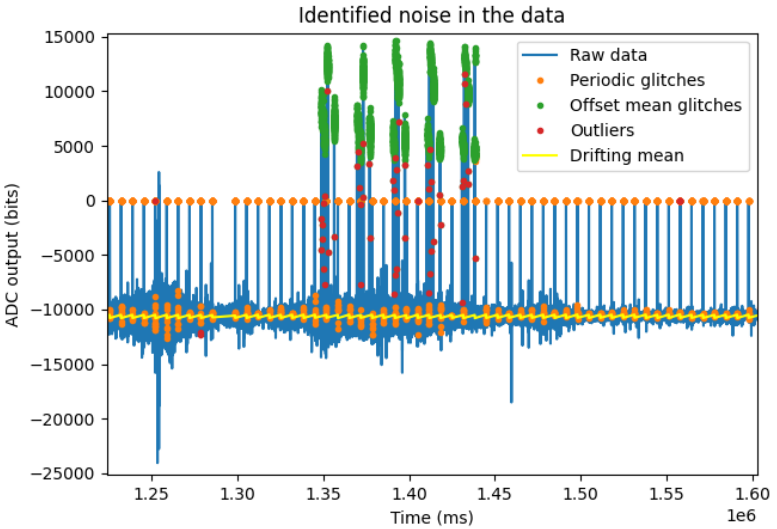


Figure 5.5 Data from the geophone, with several types of signal errors marked.

Synchronization. For data collection, the geophone and camera units need to be synchronized in some way, such that they can be chronologically aligned afterwards. The ideal way to do this is by synchronizing each unit to a real-time clock (RTC) using an internet connection. The network connection in Makulu proved very unreliable, however, so instead the units were synchronized to each other directly. In the first samples, this was done by standing next to the geophone and jumping, providing strong signals visible in both sensors. The arms were held out before the jump, and lowered after, so the time of the jump would be clear, even if the actual jump was not captured in a frame. Later, an automatic synchronization was devised, where one unit provided a WiFi hotspot that the other connected to. After doing so, a ping was sent back and forth, and from this, the round trip time and the difference in the internal clocks were calculated. Since the internal clocks counted at different rates (drifted) compared to each other, the units had to be resynchronized periodically. This was done every hour. This solution was limited by the WiFi range, so the units could not be further away from each other than a few tens of meters. However, moving them further away than this was necessary when we wanted to record data in locations where there was no fence. The solution was to bring the units close to each other and synchronize them in the beginning of the data recording, and then compensate for the drift. It was found on the last 6 data collection occasions by the lodge that the drift was relatively stable, between 149.2 and 158.1 ms/hour. The drift was also seemingly constant throughout each occasion. To examine how much of a difference external temperature made, an experiment was conducted, where data

collection was started, and one of the units was placed in room temperature for 30 minutes, then in a freezer box for 30 minutes, then again in room temperature for 30 minutes, and finally again in the freezer box for 30 minutes. It was found that the drift was very mildly affected by the ambient temperature, the difference was around 10 ms/hour. The drift was compensated with a value of 153 ms/hour. Assuming the drift stayed this constant, the error in the last two occasions would be at most around $6 \cdot 10 = 60$ ms. Since no elephants were seen on these occasions, the precise reliability of this turns out not to have been very important. Some earlier datasets appear to have had a different drift, and may have used a different unit.

Labeling. The data labeling process assumes synchronization has already been carried out. The process is mostly automatic, but requires creating a camera metadata file for each occasion in which the following information is denoted:

1. The time when (if) the sun goes down
2. The starts and ends of the segments when:
    - An elephant is visible
    - An elephant is walking
    - An elephant is within the inner circle

The times are marked by the file and frame numbers, so for example the sun can go down in file 5, frame 1037. The frames are chosen so they contain the events they mark, so for example, if the elephant is still in frame 5 and is walking in frame 6, it has started walking between frame 5 and 6, and we denote frame 6 as the beginning of the walking segment. The labeling algorithm then constructs the sets of times when:

- G_R: the geophone is recording
- C_R: the camera is recording, and the sun is up (translated to geophone time)
- E_V: an elephant is visible (translated to geophone time)
- E_W: an elephant is walking (translated to geophone time)
- E_I: an elephant is inside of inner circle (translated to geophone time).

From these sets, the next step is to construct the sets of times when:

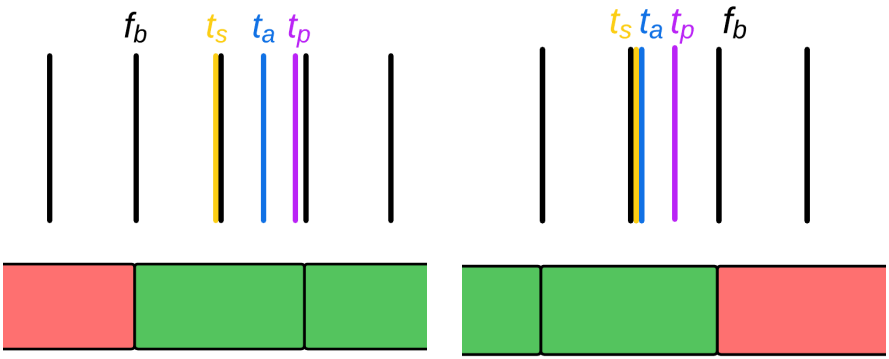
- G_S: there is supervised data; $G_R \cap C_R$
- E_P: the elephant is present; $G_R \cap (E_V \pm 10$ minutes)
- E_{WI}: the elephant is walking inside of the inner circle; $E_W \cap E_I$

The final sets can be found as:

- Negative set; $G_S \setminus E_P$
- Positive set; $E_{WI} \cap G_R$ (supervision is implied)
- Maybe positive set; $E_P \cap G_R$

We then step through the data files, taking out 2-second segments with a stride of 2 seconds. They are labeled as positive if they are completely within the positive set, and negative if they are completely within the negative set. If they are completely within the "maybe positive set" set but not positive, they are labeled as half-negative. The reason for using a 2-second stride is to avoid overlapping samples, which could end up in different sets when the data is split into training, validation, and test sets, i.e. data leakage. Note that two neighboring samples are still not completely independent, as the process generating the signal is clearly autocorrelated, with a very slowly varying background noise profile.

The earlier theses on this topic made estimates of ground wave propagation speeds, and found them to be around 160-165 m/s [26] [5]. Our work has not made any such measurements. Assuming that the wave speeds are similar in the locations in Makulu, we find the delay between a footstep being planted within the inner circle, and the seismic wave reaching the geophone, to be up to around 100 (m) / 160 (m/s) $\simeq$ 0.6 (s). This affects the timing of the samples and is thus relevant for labeling. We inspect how the delay and camera sample rate affects when the geophone signal begins and ends being positive. We assume here that the steps happen frequently enough, so that samples occurring between the first and the last step being taken always contain a full step, so we can only have problems with the first and last samples during each segment where the elephant is walking. When the elephant begins walking, we can imagine that the first positive sample could be too early to contain the first step (and thus, also no subsequent step). When the elephant stops walking, we can imagine that the last positive sample could be too late to contain the last step (and thus, also no preceding step). We analyze these two cases. For the first positive sample, we note that the worst case would be if this sample starts (and ends) as early as possible, and that the first footstep occurs as late as possible, with maximum delay. This worst case is shown in Figure 5.6a, and we see that the first footstep occurs within the first positive sample. The error margin here is 50 ms. For the last positive sample, we note that the worst case would be if this sample starts as late as possible, and that the last step occurs as early as possible with minimum delay. This worst case is shown in Figure 5.6b, and we see that the last footstep occurs within the last positive sample. The error margin here is one second which is larger than in the beginnings, so to be robust against calibration error, the positive samples have been extracted stepping from the end of each segment and moving backwards.



(a) Visualization of the worst case for the starts of walking segments.

(b) Visualization of the worst case for the ends of walking segments.

Figure 5.6 Visualizations of the worst cases when labeling samples. The black bars are the times of camera frames, and the red and green boxes are negative and positive samples respectively. t_s is the time of a step, t_a is the time when the signal from this step arrives, and t_p is the time when the signal from this step passes (defined as 350 ms after it arrives).

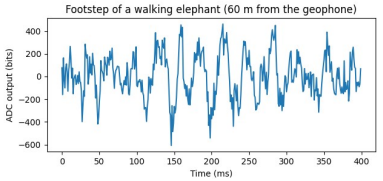
Preprocessing

Balancing. The data has been sampled naturally, so it should give an estimate of the natural sample distribution, and thus also class distribution (conditioned on the locations). The balance is presented in Table 5.1.

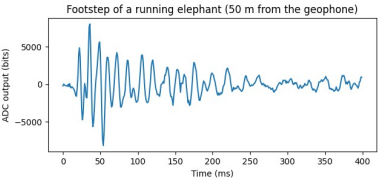
Negative class	Positive class (elephant footsteps)
116994	106 (0.09%)

Table 5.1 Class balance. As can be seen, the negative samples far outweigh the positive.

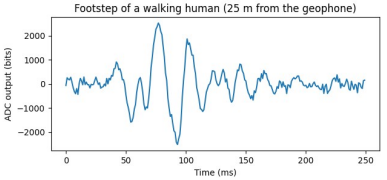
To compare this to something, we can look at when the elephants were in the same position as the geophone, treating their passing by the sensor as a Poisson process. Elephants passed on the 31st of October and the 20th of November, and they broke the sensor on the 26th of November. Assuming no elephant visits were missed, and adding the data collection date of the 8th of November, we find the wait times 8 days, 12 days, and 6 days. This gives an estimate of the intensity of 0.115 events per day, with a standard error of 0.066. If each elephant visit is around 20 minutes, this means that we should expect 1.0-3.6 minutes of elephant per day, or 24 hours of data. Since we have 20 minutes of elephant time out of around 62 hours of data, it appears like we have an over-representation of elephant data. On the other hand,



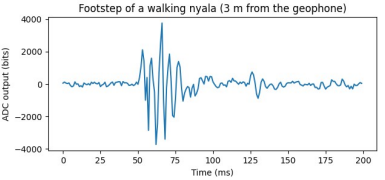
(a) Waveform of a footstep from a walking elephant. The cadence appeared to be around 1.8 Hz.



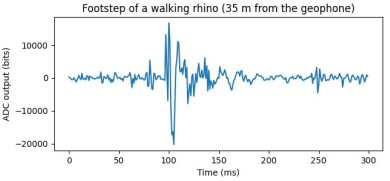
(b) Waveform of a footstep from a running elephant. The cadence appeared to be around 1 Hz, but probably only the front leg steps were counted.



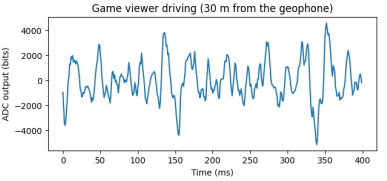
(c) Waveform of a footstep from a walking human. The cadence appeared to be around 2 Hz for individual 1 and 1.8 Hz for individual 2.



(d) Waveform of a footstep from a walking nyala. The cadence appeared to be around 2.7 Hz for the female and 2.3 Hz for the male.



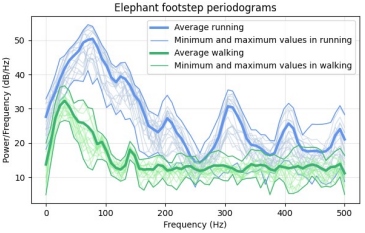
(e) Waveform of a footstep from a walking rhino. The steps were taken intermittently, with pauses for inspecting the ground.



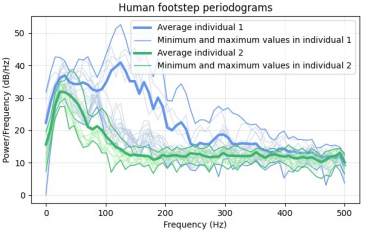
(f) Waveform from a rolling vehicle.

Figure 5.7 Waveforms from different sources. Cadence has been estimated from the geophone signals, and is defined as individual footfalls per time.

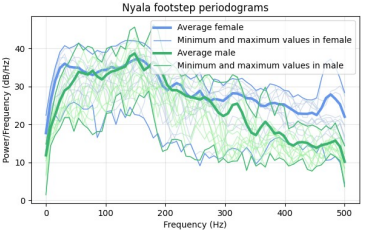
we surely have an over-representation of other difficult events, such as human footsteps and vehicles, as most data collection was done on the edge of the lodge area. Therefore, it is not certain how the data should be rebalanced, to make it as representable of the true distribution as possible. It can be assumed that if training for accuracy, this dataset has an acceptable distribution. However, when training for precision (or F1-score), it may result in a model that too strongly favors predicting the majority class. There are special splitting criteria for imbalanced classes (for example the Hellinger measure [33]), however, implementing and testing these has not been deemed feasible. Instead, attempts have been made to use data augmentation and class weights to achieve balance. For augmentation, SMOTE has been used. This is likely not ideal, as it performs linear interpolation in the feature space,



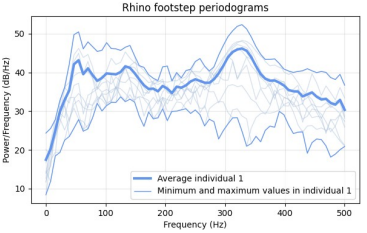
(a) PSD estimates of footsteps from an elephant. 20 running (40-60 m away) and 16 walking (50-60) have been included. There is a visible overtone structure in both estimates.



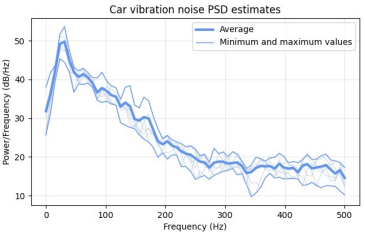
(b) PSD estimates of footsteps from humans. 19 from one individual (15-25 and 2-5 m away) and 11 from another (15-20 m away) have been included.



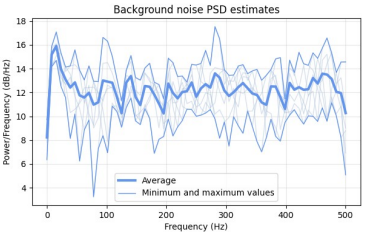
(c) PSD estimates of footsteps from nyalas. 12 from a male (2-5 m away) individual and 12 from a female individual (1-3 m away) have been included.



(d) PSD estimates of footsteps from a rhino. 10 footsteps 30-40 m away have been included. The PSD values seem improbable, putting into question whether the footsteps are from the rhino or from some unseen source.



(e) PSD estimates of vibration noise from a car rolling 30 meters away from the geophone.



(f) PSD estimates of background noise.

Figure 5.8 PSD estimates for several different footsteps. The longest footstep events seem to be up to 400 ms long, so we have used windows of this length. The estimates were produced using Welch's method with Hanning windows of length 128, and overlap 64. This method reduces variance at the expense of frequency resolution. The signal was zero-padded to the closest multiple of 64 .

and not all of the features are linear. More appropriate methods would be the ones described in Subsection 5.2.

Signal normalization. For numerical reasons, the signal has been normalized to have an amplitude on the scale of 1. This means a scaling of $\frac{1000}{2^{23}-1}$ on the signal values.

Feature extraction. Since the collected dataset is quite small, too many features could cause our model to overfit. To have an idea of what features could be useful, we study several footsteps in Figures 5.7. As can be seen, the signal waveform caused by footsteps from the same elephant can look very different. The elephant's footsteps are not easily distinguishable from the others' when visually inspecting their waveforms. To examine a bit deeper, we make estimates of the power spectral densities (PSDs), as seen in Figure 5.8. Here, we can observe that the footsteps seem to have a quite broadband spectral content when the footsteps are placed close to the geophone, or further away but powerfully. Over larger distances, however, the higher frequency content appears to be attenuated, leaving a signal with mostly low-frequency content. Separating the footsteps may be easier up close, as the spectral content is more dissimilar. Combining the time and frequency domain features shows potential for detection. For example, the walking elephant and human footsteps in Figures 5.7a and 5.7c look very different, but seem to have similar spectral content, with different amounts of concentration in the time domain. Thus, combining spectral information with other statistics, such as peak amplitude or similar, could provide sufficient information for correctly classifying the signals. Although most commonly used for sound, mel-frequency cepstral coefficients (MFCCs) are a compact and relatively noise-resilient way of representing spectral information, at the cost of some detail. The mel-scale is linear below 1000 Hz, so for all of the frequencies present in the geophone data. Appendix E shows how the MFCCs can be used to reconstruct the PSD of a footstep. Thus, we arrive at the following set of features:

- Energy (sum of signal measurements squared)
- Standard deviation (STD)
- Inter quartile range (IQR)
- Range
- Kurtosis
- The 13 first MFCCs with 40 mel-bands

This gives 18 features in total. They can be seen as a fusion of the signal feature method and average footstep method from the thesis written by Goderik and Westlund [5]. As the samples are relatively long (2 seconds) compared to the events (up

to 400 ms), we may get a low signal-to-noise ratio, which could make detecting the weaker footsteps difficult. A solution to this could be to calculate the features on smaller windows of the sample and aggregate them, or choose one of the windows through some metric, and calculate all the features on just the signal in this window. On the other hand, several footsteps may occur in the same sample. This could decrease the variability of the features. The PSD estimate of elephant footsteps when walking in Figure 5.8a was redrawn with a sample length of 2000 and examined. The resulting estimate was slightly lower in energy, but had a lot less variability. The overtone structure was much more clearly visible. Thus, we have proceeded with calculating the features over the whole samples.

The correlation coefficients between the features have been calculated, and are presented in Appendix B. It was found that some of the MFCCs were highly correlated in groups, indicating that there may be a more compact way to represent the spectral information. To avoid including redundant information, principal component analysis (PCA) dimensionality reduction can be used. This has been tried out, as is described later in this section. Most of the principal components were initially kept, allowing the feature selection to choose how many to keep.

Feature normalization. All features have been Z-score normalized, i.e. normalized to zero-mean and unit standard deviation. In the cases where PCA was applied to the MFCCs, this was done after the normalization. Note that using normalized features should not matter for a tree-based model.

Decision tree modeling

All modeling was done with a random seed of 1, for reproducible results. The modeling focused on decision trees, foregoing random forests and gradient-boosted trees. Between the ways of choosing splits ("best" and "random"), the "best" splitter was employed. It selects the optimal split, maximizing performance at the cost of a little extra training time. Between the entropy measures "Gini Index" and "Information Gain", the Gini index was chosen, as it is markedly faster to compute, while very rarely producing different splits than the information gain [34]. This makes them practically equivalent, and it should be noted that when they do select different splits, the one produced by the information gain measure is not necessarily better. In the modeling, we wished to compare and evaluate two things. The first was which parameter set to use, and the second was data balancing methods.

The first step of the modeling was to find out how the feature sets compared. To do this, Bayesian Optimization (BO) with 100 iterations, and 10-fold stratified cross-validation (10-SCV) in each hyperparameter point, was carried out on the data with all 18 features, and no PCA. The parameter ranges were:

- Max depth (1-50)
- Min samples in a split (2-20)
- Min samples in a leaf (1-20)
- Max features for a split (1-25)

The found optimal hyperparameters were then used to do Backward Feature Selection (BFS) using 10-SCV to evaluate the performance of each feature set. This was carried out on both the features with and without PCA applied. The performance of the models using the two found reduced feature sets was compared, and the best-performing reduced feature set was kept for the next step. No balancing was applied here.

In the second step, we looked at balancing methods, and evaluated the performance of all combinations of the following "hyper"-hyperparameters:

- All features without PCA vs. the reduced feature set found in the last step
- Balancing using class weights vs using SMOTE vs. no balancing

Here, we wanted both a point estimate and a confidence interval. To achieve this, we performed 10-SCV, calculating the mean and standard error of each metric over the folds. The model we trained on 10-1=9 folds, and used to make predictions on the remaining fold, was the combination of Bayesian hyperparameter optimization and a decision tree. Thus, for every cross-validation iteration, we ran BO for 30 iterations on the 9 folds, and used the final best model to make predictions on the last fold. The BO also uses 10-SCV internally. Viewing hyperparameter optimization as part of the model in this way is common in a nested cross-validation setting, and allows for an unbiased estimate of the performance of the full model, including hyperparameter optimization. The same parameter ranges as before were used.

As a bonus, a decision tree is compared to a random forest and a gradient-boosted tree, both trained with 50 estimators, as well as the industry-standard algorithm XGBoost (also a sort of gradient-boosted tree). They all use the reduced feature set and no balancing. The random forest and gradient-boosted tree use the hyperparameters found before the feature selection, while XGBoost uses default parameters. The random forest uses the square root of the total number of features in each tree, and the gradient-boosted trees use a learning rate of 0.01 and the Friedman MSE splitting criterion. They are evaluated on 10-fold stratified cross-validation.

5.9 Results

Denoising

An example result of the denoising algorithm is seen in Figure 5.9. It has been verified, by inspecting the denoised data, that it is very seldom that a real event in the signal is affected by the algorithm. The denoising is not successful on all files, so three files have been excluded from the denoised data.

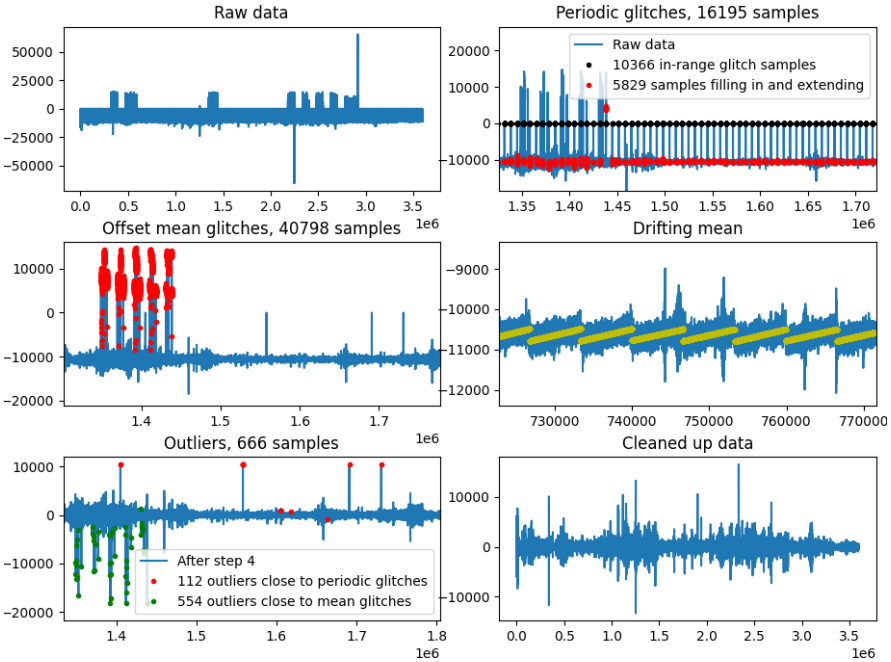


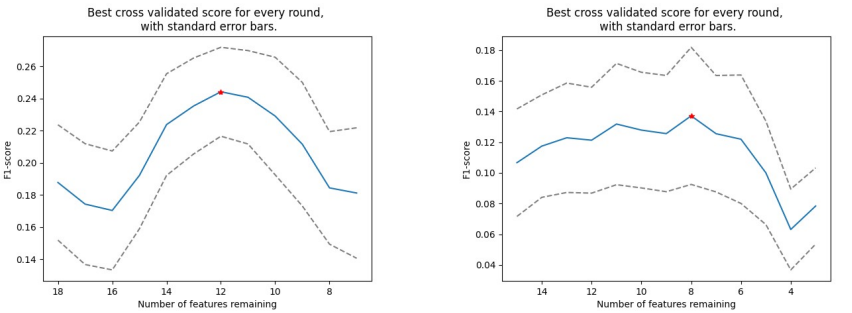
Figure 5.9 A result from the denoising algorithm. No free-standing glitches are seen in this figure.

Model

The found hyperparameters from the initial BO are:

- Depth: 23
- Max features in a split: 17
- Min samples in a leaf: 2
- Min samples in a split: 4

These hyperparameters also reflect the values seen in the best tree (for example, it is 23 layers deep), which is not necessarily true, as they only provide bounds on the possible values in the actual tree. The BO was carried out on a training set comprising 90% of the data, and these hyperparameters had a cross-validated F1-score of 0.22. On the hold-out 10%, the model did not manage to make any correct positive predictions, however, this is a rather unreliable evaluation of generalization performance. The found hyperparameters were used in the feature selection step. We now proceed to the feature selection. The progression of the best F1-score when no PCA was applied is seen in Figure 5.10a.



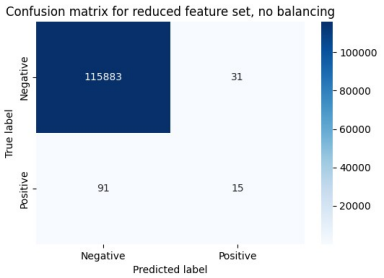
(a) BFS progression with no PCA applied. **(b)** BFS progression after applying PCA.

Figure 5.10 The feature selection processes.

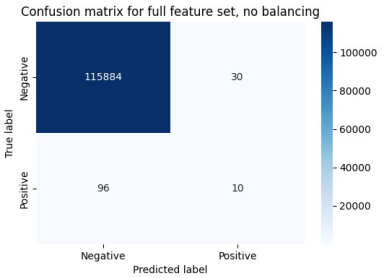
The best result was found with the 12 following features (in no particular order):

- STD
- IQR
- Range
- Kurtosis
- $\text{MFCC}_{0,1,2,4,5,8,10,11}$

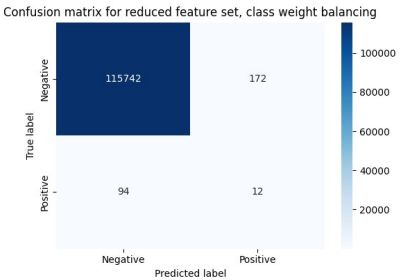
The features were thrown away in the order MFCC_{12}, energy, MFCC_9, MFCC_3, MFCC_7, MFCC_6, STD, MFCC_0, MFCC_2, MFCC_4, range. The progression after applying PCA is seen in Figure 5.10b. We start with the 10 most important principal components. We see that the results are generally worse than before, which indicates that the model is less able to create effective splits with the principal components. The order of removal was $\text{MFCC}_{\text{PCA:10}}$, $\text{MFCC}_{\text{PCA:4}}$, range, $\text{MFCC}_{\text{PCA:9}}$, energy, kurtosis, $\text{MFCC}_{\text{PCA:2}}$, $\text{MFCC}_{\text{PCA:3}}$, $\text{MFCC}_{\text{PCA:7}}$, STD, $\text{MFCC}_{\text{PCA:6}}$, $\text{MFCC}_{\text{PCA:8}}$.



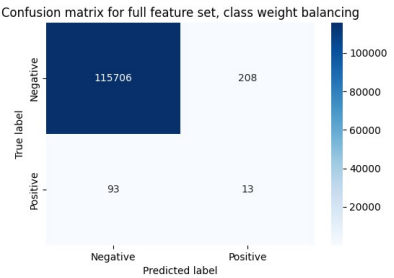
(a) Confusion matrix for the reduced feature set and no balancing.



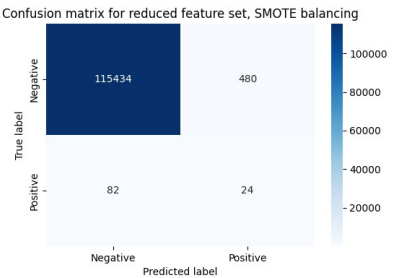
(b) Confusion matrix for the full feature set and no balancing.



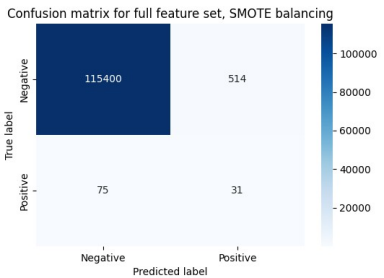
(c) Confusion matrix for the reduced feature set and balancing using class weights.



(d) Confusion matrix for the full feature set and balancing using class weights.



(e) Confusion matrix for the reduced feature set and balancing using SMOTE.



(f) Confusion matrix for the full feature set and balancing using SMOTE.

Figure 5.11 Confusion matrices of aggregated classification over all folds.

The order in which the principal components were removed does not seem to show that principal components capturing high variance were more useful for explaining the data. The best result was found with the following 8 features:

- STD
- IQR
- $\text{MFCC}_{\text{PCA:1,3,5,6,7,8}}$

Note that the PC numbering is from 1 to 13. All in all, the results seem rather unreliable due to the large standard error, although there seems to have been some benefit from getting rid of a few features, in the case when no PCA was applied.

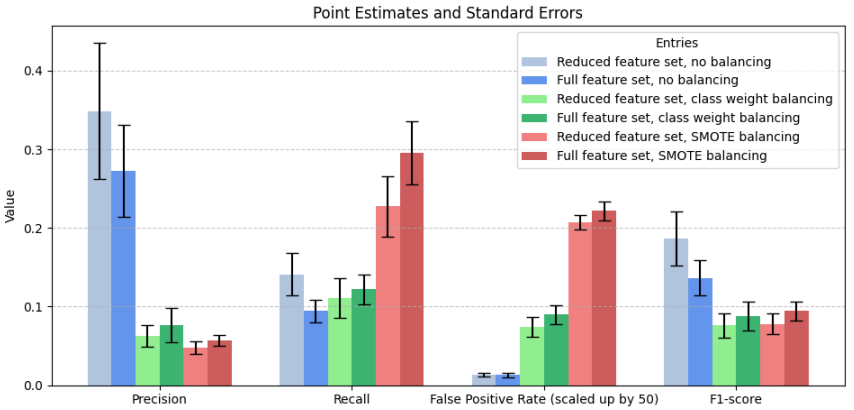


Figure 5.12 Mean values and standard errors of performance measures over the 10 outer folds, with 30 iterations of Bayesian optimization, internally using 10 folds.

We now move to the balancing methods, keeping the best reduced feature set from the feature selection without PCA. The (aggregated over all folds) confusion matrices from the 6 setups have been shown in Figure 5.11, and a bar chart of the generalization performance is seen in Figure 5.12. As the latter figure shows, there is high variability in the results, but it is also clear that the best-performing models are those without balancing. Between those, the one using the reduced dataset appears to have a non-significant edge. We are seeing a slightly worse F1 score for the model using the reduced feature set and no balancing, than we did in the feature selection. However, the best cross-validated F1-score seen in the feature selection (0.244) is not significantly different from the one seen here on a p=0.05 level (t-distribution, 10 DoF). To be significantly different on this level, it would need to be outside of the interval $[0.110, 0.263]$. Furthermore, as noted by Bates et al., the standard error is often an underestimation of variability in a cross-validated score [35]. We can also find some potential explanations for worse performance here. It may be due to the free hyperparameters not finding a good setting in 30 iterations of BO, thus not benefitting from the hyperparameter optimization. The

result seen in the feature selection is also biased to be slightly too large, as we choose the feature set with the highest value (this can be understood through order statistics). On the other hand, the result from the nested BO also has a slight bias to be too low, as the fold removed for testing in the cross-validation wrapping the BO means we have 10% less data to train on than we did in the feature selection.

Looking at the false positives for the model with the reduced future set and no balancing, we plot their occurrences in Figure 5.13. We see that they are not evenly distributed, so they are seemingly not occurring independently. Investigating the data from the times when the false positives occur, we find that they appear to happen when there are humans around in the camera footage, and footsteps are visible in the geophone data.

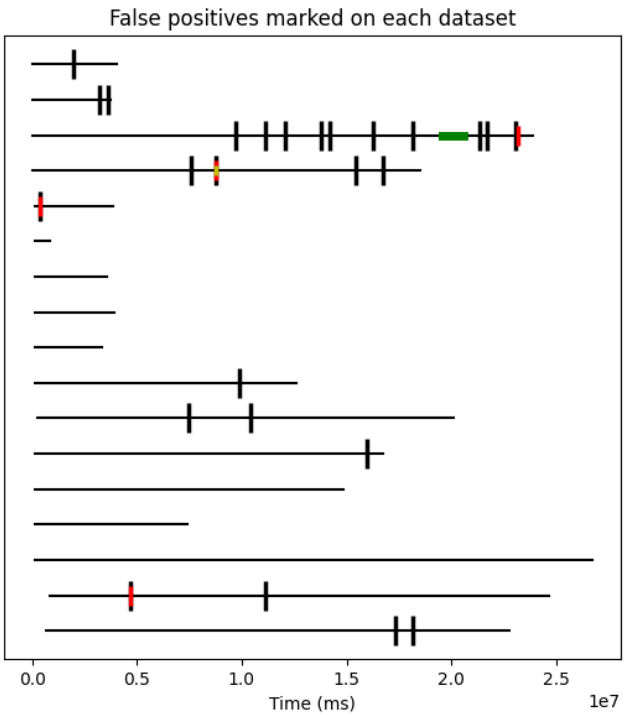


Figure 5.13 False positives marked on each data collection occasion. The green bar shows the segment containing positive samples. The red bars mark false positives within 120 seconds of the last, and the yellow bar is the third false positive within 120 seconds.

Finally, the bonus results of comparison between the decision tree, random forest, and gradient-boosted trees are found in Table 5.2. No big conclusions can be drawn from this, as the random forests and gradient-boosted trees have not been hyperparameter-optimized. However, we can note that both versions of gradient-boosted trees show potential to improve upon the performance of the simple decision trees. The XGBoost classifier was noted to train very quickly, which enables extensive hyperparameter optimization as well as performance evaluation.

	Precision	Recall	FPR
Decision tree	0.44 ± 0.079	0.18 ± 0.021	$2.5{\cdot}10^{-4} \pm 4.5{\cdot}10^{-5}$
Random forest	0.3 ± 0.14	0.029 ± 0.014	$1.7{\cdot}10^{-5} \pm 1.6{\cdot}10^{-5}$
Gradient-boosted trees	0.47 ± 0.095	0.16 ± 0.023	$2.2{\cdot}10^{-4} \pm 5.1{\cdot}10^{-5}$
XGBoost	0.62 ± 0.14	0.14 ± 0.039	$4.3{\cdot}10^{-5} \pm 1.8{\cdot}10^{-5}$

Table 5.2 Comparison of tree-based classifiers. The values are reported as: mean $\pm$ standard error of the mean.

5.10 Analysis

Analysis of results

The main limiting factor for optimizing and evaluating the footstep classifier was the amount of available positive data. The detection problem appears to have some complexity, so a lot of data is necessary to find a robust model with good generalization performance. The high variability in model performance based on how the dataset was split into training and test sets, as seen in Section 5.9, shows that the dataset does not cover the sample space sufficiently. The occurrences of false positives indicate this. They tend to be caused by human footsteps. Since there are many human footsteps in the positive data, the model has likely learned that their signals are good indicators for positive samples. With more positive data, the apparent correlation between human and elephant footsteps occurring in the samples will disappear, which should give a model with much better generalization performance. Using the F1 score to train on turned out to be problematic with this small number of positives. As we were sometimes predicting only 1 or 2 positive samples, the score became very volatile. Finally, although likely not an issue, the available positive data was from the early, manually synchronized sets of occasions, so the reliability of the labeling is not perfect.

It should also be noted that the positive data, that was available, was from a single occasion; a single location, a single elephant, a single weather condition, etc. Due to this, these samples are correlated, and not independently drawn samples from the distribution of signals from any time, any position in Makulu. There was also a

lot of background noise in these samples, making it possible that the model learns to predict based on this noise instead of the actual footstep signal. This also limited the possibility of augmenting the data by using the linearity of ground vibrations, and adding empty background samples to the positives, as the signal-to-noise ratio was already low.

It is unsurprising that balancing the data, by upsampling the (minority) positive class or increasing its weight, increases the probability that the model predicts this class, leading to more false positives. Since there is no classification threshold in a decision tree, it is crucial to get the hyperparameters right to get the desired balance between recall and false positive rate. More experimentation with balancing may lead to more desirable performance.

Using a meta-model to improve performance

In a real-use setting, we are interested in detecting elephant presence, and whether an individual sample contains footsteps from an elephant (or a poacher) is not as interesting. For example, if we correctly detect the footstep of an elephant in a sample, the following sample contributes little information; if the elephant is present at a given time, it is also present two seconds later. On the other hand, with the current false positive rates of at best 0.000267, we expect 12 false positives every day. In practice, if an elephant passes the geophone every eight day, we will get 92 false positive detections for every time we detect an elephant (assuming we detect it every time it passes). A meta-model can be used to aggregate the individual sample detections into more reliable meta-detections, and suppress the false positives. When an elephant was present in one of our samples, it was there for 20 minutes, producing 106 positive samples. This is quite a long time, and we cannot expect that an elephant will always stay this long. However, we can estimate how many samples an elephant walking through the detection radius would produce, by looking at how many samples our elephant would have produced if all its steps were in the same direction. Our elephant travels 50 meters in 49 seconds, generating 24 samples. Let us say it generates 40 samples walking through the inner circle, and takes about one and a half minutes to do this. Now, let us say that the rule for the meta-model is that we need two detections within 2 minutes to count as a true detection. We choose to use our model with the reduced feature set and no balancing from the cross-validated Bayesian optimization. We then get a recall of 0.14, and a false positive rate of 0.000267. The number of detections when the elephant passes is then $N \sim \text{Bin}(40, 0.14)$. This gives a probability of detection of

$$P(\text{detection}) = P(N \geq 2) \simeq 0.982, \tag{5.3}$$

which is our true positive rate. We now assume that positive events are rare, which they are, and that the false positives appear independently in the predictions. As previously noted, this is not the case. However, with more diverse positive data, it

will tend to be more so; and the false positive rate will be much lower. We can then approximate the false positive rate in time by treating their occurring as a continuous time Poisson process, with a time intensity of 0.00014 events/s. This is equivalent to wait times $T \sim \text{Exp}(\lambda = 0.0001335)$. We see that $\text{P}(T \leq 120) \simeq 0.0159$. We can define $K \sim G(p = 0.0159)$ as the random variable representing how many wait times we get before the one that is shorter than 120 seconds, where G is the geometric distribution counting from 0. The total wait time is then $T_{\text{tot}}|K = \sum_{i=1}^{K} T_i + T^s$, where $T^s \sim T|T < 120$. The law of total expectation then gives that the total expected wait time is

$$\mathbb{E}T_{\text{tot}} = \mathbb{E}(\mathbb{E}(T_{\text{tot}}|K)) = \sum_{k=0}^{\infty} P(K=k)\mathbb{E}(T_{\text{tot}}|K=k) = \sum_{k=0}^{\infty} P(K=k)(k\mathbb{E}T + \mathbb{E}T^s) =$$
$$\mathbb{E}T \sum_{k=0}^{\infty} kP(K=k) + \mathbb{E}T^s \sum_{k=0}^{\infty} P(K=k) = \mathbb{E}T\mathbb{E}K + \mathbb{E}T^s \simeq$$
$$\mathbb{E}T\mathbb{E}K = \frac{1}{\lambda} \cdot \frac{1-p}{p} \simeq 463678 \text{ seconds} \quad (5.4)$$

We have neglected $\mathbb{E}T^s$, as this is less than 60 seconds (the distribution is strictly decreasing). This number of seconds translates to around 5 and a half days. The simplification of the sum came from identifying the expectation value and probability function of the geometric distribution starting from 0. If we get an elephant passing by every 8 days, the practical precision is around 40%. If we make similar calculations, but this time requiring 3 detections within 2 minutes, we get

$$P(\text{detection}) = P(N \geq 3) \simeq 0.932. \quad (5.5)$$

We then get a new p for the geometric distribution, that is

$$p = P(T_i + T_{i+1} < 120) = \int_0^{120} \int_0^{120-t_i} f(t_{i+1})dt_{i+1} f(t_i)dt_i =$$
$$\int_0^{120} F(120-t_i)f(t_i)dt_i = \int_0^{120} (1 - e^{-\lambda(120-t_i)})\lambda e^{-\lambda t_i} dt_i =$$
$$\lambda \int_0^{120} (e^{-\lambda t_i} - e^{-120\lambda})dt_i = \int_0^{120} \lambda e^{-\lambda t_i} dt_i - \lambda e^{-120\lambda} \int_0^{120} dt_i$$
$$= F(120) - 120\lambda e^{-120\lambda} \simeq 0.000127. \quad (5.6)$$

Thus, the expected wait time between false positives is 58973901 seconds, or 683 days. The precision is then around 98.8%. These calculations are optimistic, since they assume independently occurring false positives, but they highlight that the model can be improved with a meta-structure, and with more data and a better model, there is potential for a very reliable system. With the current dependently

occurring false positives, we see that the three-sample meta-model would give one false positive detection in our dataset.

Further work

The clear direction of further work based on ours, is to use the established problem definition and hardware concept to gather a lot more data. A recommendation would, in this case, be to examine the options for internet connectivity for synchronization of the data streams. Furthermore, although the labeling is already fast, it could be even more automated by triangulating the target using multiple cameras.

In expanding on our work, several modifications to the problem formulation can be explored. By moving the inner ring to the outer, and defining all visually identified steps within as positive, the model can be exposed to more SNRs, and may achieve a larger range of detection. This, however, introduces some complications. Although not necessarily problematic, performance metrics for such a model will say less about how optimal the model is. More importantly, labeling will require very good supervision. As footsteps may not be detectable by visual inspection of the signal and its features, the source-to-sensor delay will need to be compensated for automatically; otherwise there may be mislabelings.This would also require good visibility and higher-quality cameras. Moving the outer ring to the inner would not give a useful problem formulation, as one ideally wants to be able to detect footsteps from as far away as possible. By labeling potentially detectable footsteps outside of the circle as negative, one would be training the model to suppress weaker footstep signals.

Another direction of exploration is automated data collection and labeling. For this, animal trackers could be used. If an animal is known to be within the inner ring and moving, samples could be collected, labeled, and stored for future retraining of the algorithm, or used for incremental learning. If edge cases are interesting, one could also collect highly informative data by having a setup with a camera and a geophone, and running a detection algorithm on the geophone. If an edge case sample, where the predictor is unsure (close to 0.5 certainty for a predictor that has such a property), is encountered, the camera can be triggered to take a photo. This combination of sample and photo is then sent to a human, who attaches a label to the sample, preferably using some easy-to-use graphical user interface (GUI).

With higher frame rate, image resolution, and accurate information on the signal ground speed, individual footsteps could also be observed and their ToA could be calculated, further aiding with and enriching labeling, enabling more use cases.

As more data becomes available, several possibilities also arise. As demonstrated in Seismic savanna, pre-trained deep neural nets have a lot of potential for accurate

predictions on seismic footstep data [30]. We note that the XGBoost model in our work (set to a low classification threshold) achieved a balanced accuracy of 87.5%, higher than the model of Seismic Savanna at up to 100 m distance. However, as noted before, the variability of our score is high, and all positive samples were from a single occasion. With more data, it may also be beneficial to undersample the negative class, and compensate for this by increasing its weight. Appropriate compensation would be possible, once enough samples have been collected to properly estimate the ratio of positive and negative in the underlying distribution. This undersampling will speed up training, and allow usage of highly iterative methods, such as bootstrapping, for approximating accurate confidence bounds. If continuing with feature-based methods, it could be worth trying out Gammatone Cepstral Coefficient (GTCCs) instead of MFCCs, as they have been seen to often provide better performance with sound classification.

6

Case 3: Fence Intrusion Detection

6.1 Background

Nearly all game reserves in South Africa are surrounded by fences to prevent poaching and protect the wildlife within. These fences are a fundamental feature of protected areas (PAs) in the country, as they not only help to safeguard high-value species like rhinos and elephants, but also prevent illegal activities such as encroachment and poaching for bushmeat and other wildlife products [36]. By restricting access, fences contribute to the preservation of biodiversity within PAs, helping to maintain stable populations of various species and reduce the risk of human-wildlife conflict [37].

In South Africa, fencing is particularly common and legally mandated for certain PAs that contain high-value or dangerous species [38]. This high prevalence highlights the importance of fences in South Africa's conservation strategy. They are not only physical barriers but also integral components of anti-poaching strategies that help deter and detect illegal entry into reserves.

Fences pose unique challenges for conservation efforts, as they are often targeted by poachers who cut or break through them to gain access to protected areas. Detecting these break-ins is critical for minimizing the impact of poaching. To improve fence security, various technological solutions are being implemented [1]. Advanced sensors, such as vibration-based and acoustic sensors, are installed along fence lines to detect physical disturbances or attempts to breach the fence. One example of such a system was the South African Draadsitter, which unfortunately never made it to the market [39]. These sensors send real-time alerts to park rangers or reserve managers, enabling a rapid response to potential threats.

Overall, fence security systems are crucial for maintaining the safety and integrity of protected areas. They play a vital role in the broader conservation efforts in South Africa by preventing unauthorized access, safeguarding biodiversity, and reducing

the occurrence of illegal activities that threaten wildlife populations.

6.2 Literature review

In the context of this research, reviewing relevant literature is essential to ensure a robust foundation and informed decision-making. The selected papers were chosen based on their direct relevance to our area of focus: Machine learning for accelerometers and/or fence breach detection. The chosen papers provide insights into methodologies, frameworks, and applications similar to our own. By critically analyzing these studies, we aim to identify best practices and recognize potential pitfalls.

Intelligent Fence Intrusion Detection System: Detection of Intentional Fence Breaching and Recognition of Fence Climbing

This paper introduces a compact hardware module designed to detect and classify activity on a fence [40]. The module utilizes a 3-axis accelerometer with an inbuilt microprocessor, framing the problem as a multiclass detection task with the classes: rattling, climbing, and no event.
The authors employ a Bayesian classifier and a state machine for dynamic classification, leveraging a selected set of features for the task. They report that wave analysis in elastic planes shows maximum dynamic changes at the center of the fence, motivating the placement of the sensor at this location. The classification process begins by distinguishing between event and no event. The authors assume that the accelerometer output is weakly stationary, based on the observation that motion near the center of the fence involves planar shifts rather than rotational movement. To detect an event, they introduce signal variation as the primary feature.
Following event detection, the system differentiates between rattle and climb. Rattle is defined by periodic movement along the Z-axis, with the periodicity determined by the fence's natural resonance frequency. Climbing, on the other hand, is characterized by the application of force at various points on the fence, resulting in a non-periodic output with significant acceleration across all axes. To isolate these activities, they construct a filter bank with two filters tailored to the frequencies observed in the fence during rain, wind, and rattling.
In summary, the paper utilizes the following features:

- Relative energy of X-axis to Z-axis
- Relative energy of Y-axis to Z-axis
- Normalized energy of band-pass filter 1 in X-axis
- Normalized energy of band-pass filter 2 in X-axis
- Normalized energy of band-pass filter 1 in Y-axis

- Normalized energy of band-pass filter 2 in Y-axis
- Normalized energy of band-pass filter 1 in Z-axis
- Normalized energy of band-pass filter 2 in Z-axis

The authors utilize a window length of 1.45 seconds with a 50% stride. The system achieves high accuracy, though the dataset is limited, comprising less than 500 seconds of total data.

Automated animal behavior analysis using accelerometer activity tags

In his master's thesis, Jakob Ahokas explores the integration of accelerometer-based activity tags with machine learning models to automate the analysis of livestock behavior [41]. He collects a new dataset on cows and goats, recording acceleration data in three dimensions. Ahokas then performs feature extraction and uses a decision tree classifier, distilled from an original random forest model, to recognize various livestock behaviors.

The classifier performs well in identifying basic behaviors, achieving high accuracy for classes well-represented in the training set. To improve accuracy for underrepresented classes, Ahokas applies SMOTE to balance the dataset. He also incorporates Hämäläinen's jerk filter [42], a technique that reduces noise in acceleration data by minimizing jerk, or the rate of change in acceleration. This results in a smoother signal and more interpretable data.

Ahokas notes that while random forests have shown strong predictive performance in similar problems, an RF model with a large number of estimators could be too memory-intensive for a microcontroller-based activity tag. Consequently, he uses a large RF model for offline analysis, extracting optimal features and hyperparameters to apply in a simplified decision tree that fits the microcontroller's constraints. Ahokas tests several time window lengths, including 10 second, 5 second, and 2.4 second windows. For feature selection, he employs raw time-domain features, dynamic component features, and frequency-domain features. Additionally, he calculates each time-domain and dynamic component feature separately for each axis (x, y, z), and for frequency-domain features, he calculates them across six frequency bands, ranging from 0 Hz to 5 Hz. This results in a total of 114 features. For feature reduction, he uses feature importance to narrow down to 20 features for the simplified decision tree. He finds that almost all frequency-domain features are irrelevant for the random forest model. Additionally, neither the 10 second nor 5 second windows improve model performance, leading him to select the more efficient 2.4 second window size.

Relevance of Papers

Each paper reviewed played a pivotal role in shaping our research direction and methodology. The key contributions from each paper are as follows:

1. "Intelligent Fence Intrusion Detection System: Detection of Intentional Fence Breaching and Recognition of Fence Climbing"
    - Inspired the placement of the accelerometer on the middle of the fence.
    - Inspired the possibility of splitting the classification problem into several consecutive parts.
    - Motivated the choice of a window length of two seconds to be valid.
2. "Automated animal behavior analysis using accelerometer activity tags"
    - Motivated the usefulness of decision tree classifiers for accelerometer data.
    - Motivated the use of SMOTE for the imbalanced dataset.
    - Motivates to not focus on frequency-domain features.
    - Motivated the choice of window length of two seconds to be valid.
    - Motivates the use of feature importance as a valid feature space reduction method.

6.3 Interviews with rangers

The full transcript can be found in F.3.

Aims and important issues

One of the most important insights to emerge from the interviews about fence intrusion systems is how fence detection systems inherently detect poachers *before* any harm can occur to wildlife. As Junel explains, this approach stands in contrast to shot-detection systems, which typically respond only *after* an incident has occurred. Tiaan emphasizes that reliability is paramount, noting that rangers currently patrol the perimeter several times a week to confirm the fence's integrity. A dependable detection mechanism, capable of identifying wire tampering—or even a single cut—would substantially reduce the need for such manual inspections, preserving the fence as a robust deterrent. Both Tiaan and Lucas additionally stress the importance of keeping sensors inconspicuous, whether by hiding them within fence poles, beneath plastic tubing, or in other discreet locations, given that poachers often locate and destroy any visible technology.

Potential problems

Beyond concealment, the system must also withstand a range of environmental and wildlife-related challenges. Rain, tall grass, and fallen branches can easily compromise fence integrity, while kudus are known to damage fences if they jump into or over them. Although elephants usually avoid fences, they occasionally knock trees

onto the perimeter, causing extensive damage. Poachers themselves employ diverse methods, including covering the wires with hollow plastic tubes, bending or cutting the fence, and even obscuring footprints with socks, diapers, or quari bush. Didi points to further hazards such as lightning strikes, insect infestations, and bushfires, which can render electronic components inoperative. Consequently, any fence detection system must be both discreet and highly resilient, able to endure severe environmental conditions while maintaining accurate performance.

Future goals

Looking ahead, interviewees acknowledge that the potential applications of fence detection systems extend beyond anti-poaching efforts alone. Junel foresees using these technologies to monitor animal enclosures, detecting attempts to escape or predator incursions. Lucas envisions integrating the systems with drones and sensor networks for real-time communication, allowing immediate responses to breaches. Didi underscores that the feasibility of these innovative solutions rests on cost and precision: a system that is too expensive or rife with false alarms may not be viable for long-term use. Lucas accordingly suggests focusing on "hotspot" areas first, given Makulu's 26-kilometer perimeter, so that resources can be directed efficiently to the most vulnerable segments. Once proven effective in these critical zones, the system could be scaled to encompass the entire reserve, paving the way for a fully integrated and sustainable defense against poaching and other threats.

6.4 Hardware and data collection

Test-site description

Data collection was performed at Makulu Sanctuary in South Africa, where the fence is constructed using three different types of poles (see Figure 6.1). Each type plays a specific role in the fence's structure and influences wave propagation characteristics.



(a) Type 1 pole: The most common pole at Makulu's fence, made of hollow plastic in a circular shape.



(b) Type 2 pole: The second most common pole, made of metal in a T-shape.



(c) Type 3 pole: A thicker, sturdier metal pole in a circular shape, acting as a barrier to wave propagation along the fence.

Figure 6.1 Different types of poles found at the Makulu fence.

Type 1 poles are the most common and are made of hollow plastic in a circular shape. Type 2 poles are metal poles built in a T-shape, with the top part facing outward from the perimeter. Type 3 poles are thicker and sturdier metal poles in a circular shape.

Type 3 poles are placed at intervals of 200 meters along the fence. Due to their size and rigidity, they act as barriers to wave propagation, reflecting mechanical waves rather than allowing them to pass through. This means that any movement along the fence will have a maximum effective propagation distance of 200 meters, confined between the nearest Type 3 poles on either side.

Between each pair of Type 3 poles, there are approximately 20 Type 2 poles and 120 Type 1 poles, spaced evenly along the 200 meters. The consistent spacing and arrangement of these poles create a regular structure that influences wave reflection and transmission along the fence.

Each pole is connected to the next with 20 evenly spaced metallic wires. These wires not only maintain the structural integrity of the fence but also affect how vibrations and waves propagate through the fence system.

Hardware

This research project utilized the SparkFun OpenLog Artemis, an open-source data logger that served as the base sensor for the fence break-in detection system (see

Figure 6.2). The device features a built-in Inertial Measurement Unit (IMU), specifically the ICM-20948, capable of capturing motion data, which was essential for this project.
The OpenLog Artemis supports a wide range of sensors through interfaces like I2C, SPI, and UART, and can log at customizable intervals ranging from 500 times per second to once every 24 hours. Sensor data is stored on microSD cards with capacities up to 32GB.
For energy efficiency, the device automatically powers down during long intervals between readings, reducing sleep current to approximately 18µA. This low power consumption allows for extended logging periods on battery power, making it suitable for long-term remote monitoring applications. Additionally, the integrated LiPo charging circuit provides a charging current of 450mA.
Figure 6.2 shows the SparkFun OpenLog Artemis used in the project.



Figure 6.2 Official image of SparkFun OpenLog Artemis product [43].

Passive data

To monitor typical fence activity without human interference, passive data was collected using the SparkFun OpenLog Artemis devices. Each device was attached to a Type 1 pole on the perimeter fence of Makulu Sanctuary using cable ties (see Figure 6.3). The sensors were positioned midway between two Type 3 poles to optimize coverage along the fence while minimizing the required number of sensors. Placing the devices on Type 1 poles also facilitated easy alignment of the sensor's orientation axes with the fence structure.
Data was logged to the internal SD card and collected the following day by retrieving the devices. For passive data collection, the devices remained in place for up to 24 hours.
Figure 6.3 and figure 6.4 show the sensor being used for passive data collection and a sample of passive data.



Figure 6.3 SparkFun Openlog Artemis placed along with a battery pack on the perimeter fence of Makulu, collecting data.

A sample of the passive data collected is shown in Figure 6.4, illustrating two seconds of acceleration measurements. This data serves as a baseline for normal fence activity against which active events can be compared.

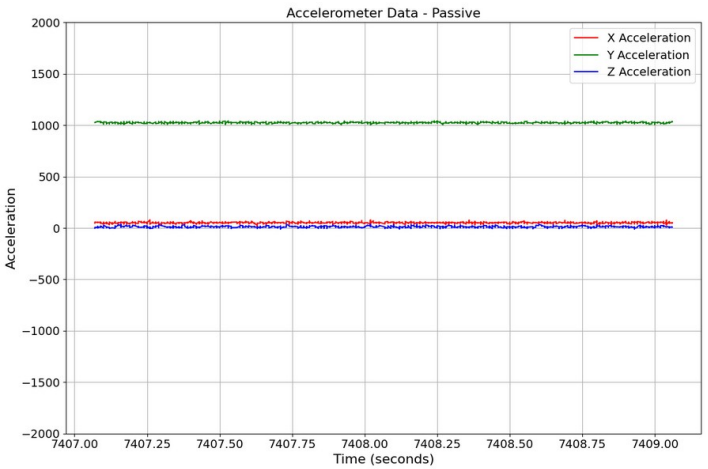


Figure 6.4 Acceleration of two seconds, measured by SparkFun OpenLog Artemis during passive data collection

Active data

For active data collection, the same setup was used as described in Section 6.4. However, in this case, deliberate events were introduced to the fence to simulate

poacher activities and animal interactions. Four accelerometers were attached to the same Type 1 pole to increase the amount of data collected simultaneously.
Active data collection was conducted at varying distances from the sensor-equipped pole to assess the detection range and signal attenuation along the fence. The distances chosen were:

- 10 meters
- 50 meters
- 100 meters

The accelerometers were synchronized by timing their activation, allowing for accurate alignment of data streams during post-processing.
Volunteers at Makulu, under the guidance of rangers, emulated the following behaviors identified in Section F.3:

1. Poacher Simulation: Bending the wires open and passing through the fence (see Figure 6.5).
2. Animal Simulation: Mimicking an animal jumping into the fence.



Figure 6.5 Volunteer simulating a poacher bending the wires and passing through the fence during active data collection.

The active data distribution used for the experiments is the following:

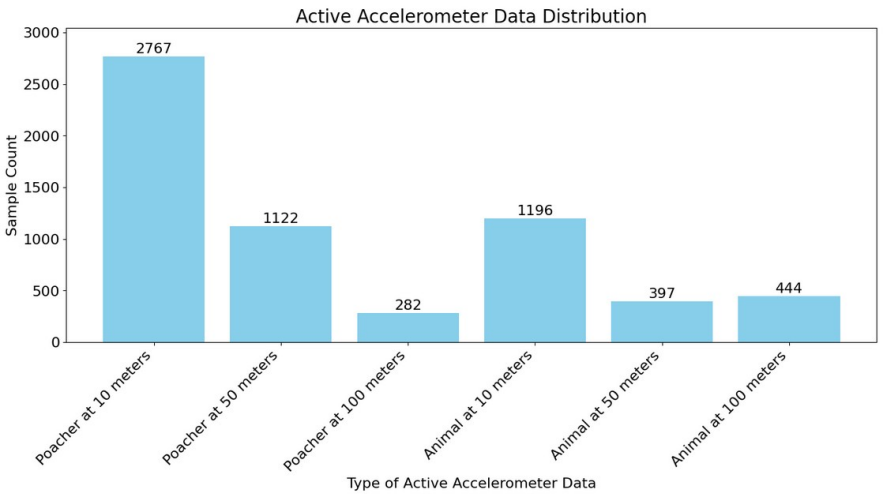


Figure 6.6 Active accelerometer data distribution

Table 6.1 summarizes the number of samples collected for each activity at the different distances.

Table 6.1 Accelerometer Sample Counts by Activity Type and Distance

Activity	10m	50m	100m	Total
Poacher	2767	1122	282	4171
Animal	1196	397	444	2037

The two plots below showcase samples collected at Makulu, one containing a break-in attempt and one containing an animal (emulation) jumping into the fence.

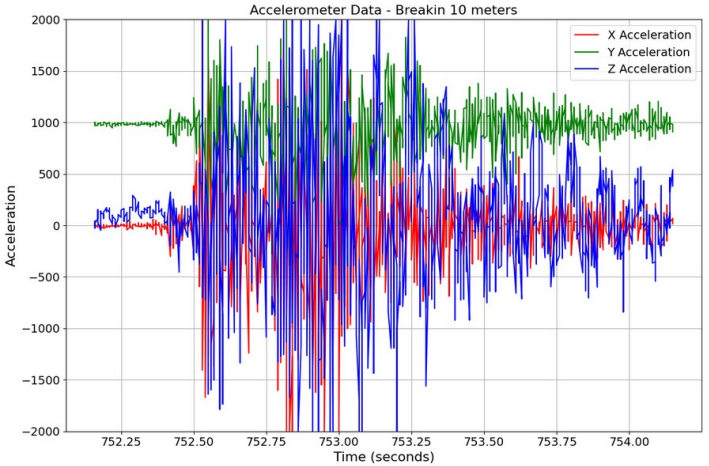


Figure 6.7 Two seconds of active data, measured by SparkFun OpenLog Artemis during active data collection: Bending the wires open and passing through at 10 meters.

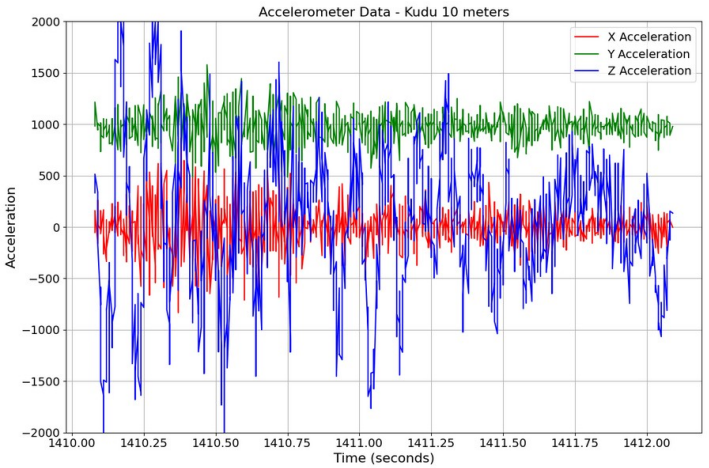


Figure 6.8 Two seconds of active data, measured by SparkFun OpenLog Artemis during active data collection: Emulating an animal jumping into the fence at 10 meters.

A real poacher intrusion occured on Makulu while one of the authors was on-site. Evidence of the intrusion (apart from camera footage of the paoachers captured by camera traps) could be found in both footprints and opened fences. Pictures of the fence and a footprint around the exit-site of the poachers can be found in Appendix L.

6.5 Implementation

Cascading

Inspired by the methodology discussed in Section 6.2, and with the objective of reducing computational overhead while maintaining high detection accuracy, we implemented a cascading classifier approach for Case 3. Both classifiers in the cascade are implemented as decision trees.

Cascading is a machine learning technique where multiple classifiers are arranged sequentially, with each designed to handle a specific aspect of the classification problem. The output of one classifier becomes the input for the next, allowing for a progressive refinement of the decision-making process. This hierarchical structure enables the system to efficiently manage computational resources by allocating more processing power only when necessary.

In our context, the top classifier in the cascade is tasked with distinguishing between passive and active samples. This initial step is relatively straightforward due to the pronounced differences in signal patterns between passive states (no significant activity near the fence) and active states (presence of any activity). By quickly filtering out passive samples with minimal computational effort, we significantly reduce the volume of data that requires more intensive analysis.

Once active samples are identified, the bottom classifier focuses on the more challenging task of differentiating between the two classes of active data: Poacher and animal. By concentrating computational resources on this classifier only when active samples are detected, we optimize the overall efficiency of the system.

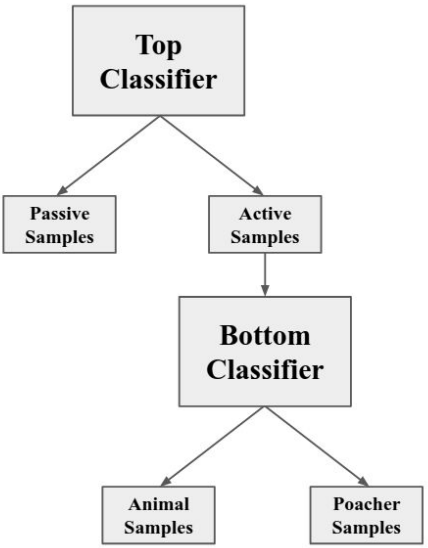


Figure 6.9 Visual representation of the cascade classifier.

The cascading approach offers several advantages:

- **Efficiency:** By filtering out the majority of samples at early stages, we reduce the computational load on subsequent classifiers.
- **Modularity:** Each classifier can be independently developed and optimized for its specific task, allowing for easier maintenance and potential upgrades.
- **Improved Accuracy:** Breaking down the classification problem into smaller, manageable tasks can lead to better overall performance compared to a single classifier.
- **Scalability:** Additional classifiers can be added to the cascade to handle more classes or to further refine the classification without overhauling the entire system.

The two classifiers, and the combined classifier, will here be referred to as the top classifier, the bottom classifier and the cascade classifier respectively.

Preprocessing

Window size. Recordings were split into windows of 2 seconds, with 1 second stride. In other words, each window has 50% overlap with the neighboring windows, to make sure that no information is lost between samples. Active data was plotted and labeled using a python script.

Data splitting. The procedure described in 4.5 was used.

Balancing. SMOTE was used to balance the training set. Oversampling and undersampling were also tested but did not improve results.

Features

For detecting fence break-ins, a variety of time-domain features were chosen based on recommendations of the paper summarized in 6.2, and intuition. Each feature is calculated for each axis (X, Y, Z), and a brief explanation is provided below.

1. **Mean (X, Y, Z)**: The average value for each axis over the window, indicating the general direction or offset of acceleration.

2. **Mean Absolute Deviation (MAD) (X, Y, Z)**: The average absolute distance from the mean, capturing variability without squaring.

3. **Standard Deviation (X, Y, Z)**: Measures the spread of values around the mean, where larger values indicate more variability.

4. **Variance (X, Y, Z)**: The squared spread of data points around the mean, providing another measure of variability.

5. **Signal Magnitude Area (SMA)**: The combined magnitude of all three axes, representing the overall intensity of movement.

6. **Root Mean Square (RMS) (X, Y, Z)**: The square root of the mean of squared values, reflecting the energy content in each axis.

7. **Correlation (XY, XZ, YZ)**: Measures the linear relationship between pairs of axes, indicating if axes move together or inversely.

8. **Interquartile Range (IQR) (X, Y, Z)**: The difference between the 75th and 25th percentiles, indicating the spread of the middle half of data.

9. **Range (X, Y, Z)**: The difference between the maximum and minimum values, showing overall amplitude of movement in each axis.

10. **Peak Amplitude (X, Y, Z)**: The largest absolute value recorded, identifying the most extreme movement or event.

11. **Mean Cross Rate (X, Y, Z)**: The frequency at which the signal crosses its own mean, capturing changes in direction or oscillation.

12. **Kurtosis (X, Y, Z)**: Indicates how heavy or light the tails of the distribution are, helping detect outlier-prone or extreme behaviors.

13. **Skewness (X, Y, Z)**: Measures the asymmetry of the data distribution, showing whether data tend to be higher or lower than the mean.

14. **Amplitude Changes (X, Y, Z)**: The standard deviation of differences between consecutive samples, capturing short-term fluctuations or jitter.

Hard Negative Mining

The implementation is identical to that described in 4.5.

Feature importance

The implementation is identical to that described in 4.5, except that the subjective feature importance threshold was set to 0.01%.

Sequential Feature Selection

The implementation is identical to that described in 4.5.

Bayesian Optimization

The implementation is identical to that described in 4.5.

6.6 Results

In this section, a range of results are presented. The results for each model include a confusion matrix and a set of quantitative metrics. For a refresher on the definition of the various metrics, we refer to Section 2.4.
For the sake of brevity, the results presented here will be restricted to the base cascade classifier and the optimized cascade classifier. The base cascade classifier uses all features for both the bottom and top classifier. The optimized cascade classifier uses an optimized feature set for both the top and bottom classifier, without hyperparameter optimization.

Base model

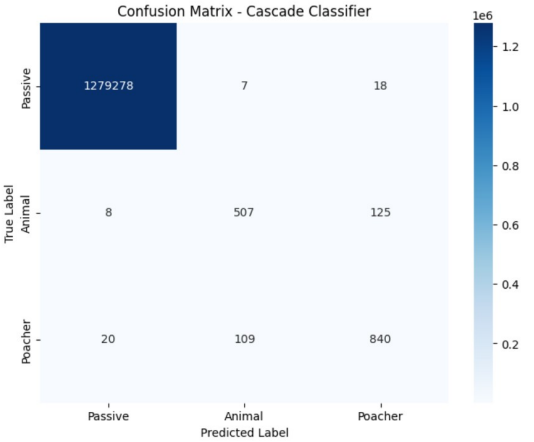


Figure 6.10 Confusion matrix for fence intrusion detection using all features

As per usual, we present the quantitative metrics of the target class - in this case "Poacher".

Metric	Value
Accuracy	0.9998
Precision	0.8545
Recall	0.8669
F1-Score	0.8607

Table 6.2 Quantitative metrics for fence intrusion detection using all features

Optimized feature selection

For both the top and bottom classifier, an optimal feature subset was found using feature importance. The intermediate results and iterations of finding an optimal feature subset for each classifier is presented in Appendix I.2
For the top classifier the following optimized features are used:

1. Range X
2. Mean Cross Rate Z
3. Mean X
4. Mean Y
5. Peak Z

6. Signal Magnitude Area (SMA)
7. Mean Z

For the bottom classifier the following optimized features are used:

1. Kurtosis X
2. Mean Cross Rate Z
3. Signal Magnitude Area (SMA)
4. Mean Cross Rate X
5. Mean Cross Rate Y
6. Root Mean Square Z (RMS Z)
7. Mean X
8. Correlation YZ
9. Mean Y
10. Interquartile Range Z (IQR Z)
11. Correlation XY
12. Root Mean Square X (RMS X)
13. Peak Z
14. Standard Deviation Z

Using these feature sets for the optimized cascade classifier yielded the following results:

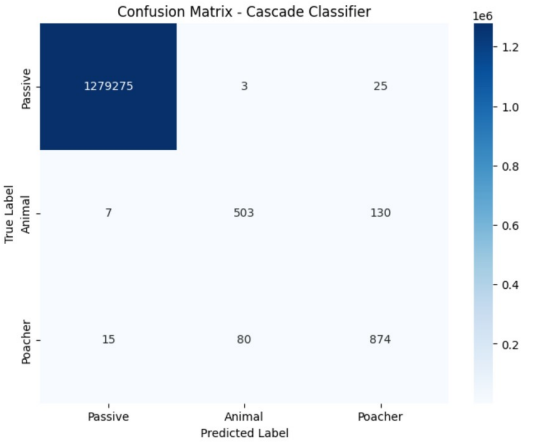


Figure 6.11 Confusion matrix for fence intrusion detection using feature importance for top and bottom tree

As per usual, we present the quantitative metrics of the target class - in this case "Poacher".

Metric	Value
Accuracy	0.9998
Precision	0.8494
Recall	0.9029
F1-Score	0.8753

Table 6.3 Quantitative metrics for fence intrusion detection using feature importance for top and bottom tree

6.7 Analysis

Results analysis

We remind the reader that the primary objective of this research project is to minimize the FPR and consequently maximise precision. To this effect, the model performs quite well. The overall performance indicates that the classifier is highly effective at identifying passive samples (only 28 misclassifications out of over 1.27 million samples) but faces some challenges in accurately distinguishing between animal and poacher events.

The classifier correctly identifies 874 poacher samples but misclassifies 80 as animal and 15 as passive. While the classifier performs reasonably well in detecting poacher events, the significant number of misclassifications suggests that there is

room for improvement. The confusion between poacher and animal classes is notable, similar to the observations in the animal class analysis. This may again be due to overlapping characteristics in the accelerometer data for these two classes. Additionally, some poacher activities might produce less pronounced vibrations, making them harder to distinguish from animal interactions or even passive fence states.
Since we lacked a means to verify whether passive samples were truly passive—such as a camera monitoring the fence during the night—some samples in the passive dataset were automatically labeled as passive under the assumption that no events occurred. However, as shown below, some of these samples appear to indicate activity during what was assumed to be passive data collection.

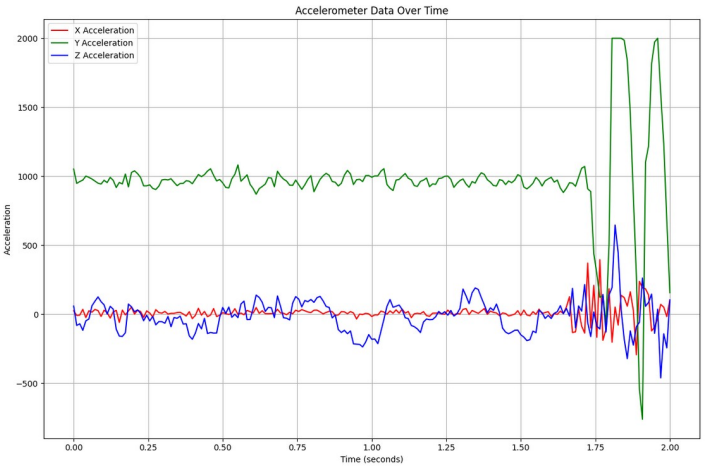


Figure 6.12 Example of "false positive". Passive sample that was classified as poacher. This sample can be seen to most likely not represent the passive class, meaning that the sample is unknowingly mislabeled. Therefore it is difficult to state whether this is truly a false positive or not.

Once again, we find that optimizing the hyperparameters of the decision tree worsens the results significantly. A likely explanation is that optimizing the hyperparameters on the training set causes the model to overfit and perform worse on unseen data.

Feature analysis

Initially, we extracted a comprehensive set of features from the accelerometer data, consisting of 13 different features calculated for each of the three axes (X, Y, Z), plus one additional combined feature (SMA), resulting in a total of 40 features. These features are listed in Section 6.5.
For the optimized cascade classifier, we reduced the number of features significantly:

- Top Classifier: Reduced from 40 to 7 features.
- Bottom Classifier: Reduced from 40 to 14 features.

This means we removed 33 features for the top classifier and 26 features for the bottom classifier, leading to more streamlined models.
Looking at the features used for the top classifier, which distinguishes between events and non-events, we find that the selected features focus on general movement characteristics, such as mean values, range, and overall movement intensity (SMA). These features are suitable for distinguishing between normal fence conditions (non-events) and any significant movement (events), regardless of the event type.
For the bottom classifier, the features are more specialized and include statistical measures like kurtosis, standard deviation, and interquartile range. The inclusion of correlation features suggests that the relationship between different axes is important for distinguishing between jumping into the fence (which might involve more coordinated movements across axes) and breaching (which may have distinct patterns). The focus on mean cross rates and RMS values indicates an emphasis on the frequency and energy of movements, which could differ between the two types of events.

Estimating deployment on SoC

Following the methodology described in subsection 4.7, we estimated the inference time of our accelerometer-based cascade classifier on the Sony Spresense microcontroller. Estimating the execution time is essential to ensure that the system can perform real-time detection of fence break-ins within the hardware constraints of the deployed environment.
After optimizing our feature selection and classifiers, we measured the following average execution times on our development hardware (specifications detailed in Appendix K):

- Feature Extraction Time (Optimized Top Classifier): ≈ 0.7 ms
- Inference Time (Optimized Top Classifier): ≈ 0.5 ms
- Feature Extraction Time (Optimized Bottom Classifier): ≈ 2.1 ms
- Inference Time (Optimized Bottom Classifier): ≈ 0.6 ms

Summing these times, the total processing time on the development laptop for both stages of the cascade (assuming the bottom classifier is triggered) is approximately four milliseconds.
To estimate the execution time on the Sony Spresense, we apply the scaling factor based on the ratio of the laptop's CPU frequency to that of the Spresense, as previously established:

$$\text{Scaling Factor} = \frac{\text{Laptop CPU Freq}}{\text{Spresense CPU Freq}} = \frac{4,200\ \text{MHz}}{156\ \text{MHz}} \approx 26.92$$

Using this scaling factor, we calculate the estimated processing time on the Spresense:

$$\begin{aligned}\text{Total Processing Time}_{\text{Spresense}} &= 4\ \text{ms} \cdot 26.92 \\ &\approx 107.7\ \text{ms}\end{aligned}$$

Thus, with the optimized feature set and cascade approach, the estimated total processing time on the Sony Spresense is approximately 108 ms, well within the acceptable range for analyzing 2 second samples in real time.
For comparison, we also measured the average execution times when using all 40 features for a single decision tree on the development hardware:

- Feature Extraction Time (Single Tree, All 40 Features): ≈ 3.6 ms
- Inference Time (Single Tree, All 40 Features): ≈ 0.7 ms

If we were to use two trees, each with all 40 features, the total processing time on the development laptop would roughly double to:

$$\text{Total Processing Time} = (0.7\ \text{ms} + 3.6\ \text{ms}) \cdot 2 \approx 8.6\ \text{ms}.$$

Scaling this to the Spresense:

$$8.6\ \text{ms} \cdot 26.92 \approx 231.5\ \text{ms}$$

Therefore, using two full sets of all 40 features for both classifiers—without any optimization—would result in approximately 230 ms of processing time on the Spresense. While still most likely acceptable for real-time applications, this is more than double the time achieved by our optimized feature selection approach.

Transitioning from a confusion matrix to the real world

While the confusion matrix and associated metrics provide valuable insights into the classifier's performance on test data, deploying the system in a real-world environment introduces additional considerations. This section analyzes the practical aspects of implementing the fence break-in detection system at Makulu Sanctuary, focusing on scalability, feasibility, and strategies to mitigate potential challenges.
Makulu Sanctuary's perimeter fence spans approximately 3km by 10km, forming a rectangular boundary with a total perimeter length of 26km. Given that each accelerometer can effectively monitor up to 200meters of fence due to the wave propagation limitations imposed by the Type 3 poles (see Section 6.4), we calculate that

130 sensors are required to cover the entire perimeter:

$$\text{Number of Sensors} = \frac{\text{Perimeter Length}}{\text{Coverage per Sensor}} = \frac{26\,\text{km}}{0.2\,\text{km}} = 130$$

Analyzing the system's false positive rate is crucial for understanding the practical implications of deploying such a system at scale. Based on the optimized classifier's confusion matrix:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

From the confusion matrix:

- False Positives (FP) for the poacher class:

  $$\text{FP} = 25\,(\text{Passive predicted as Poacher}) + 130\,(\text{Animal predicted as Poacher}) = 155$$

- True Negatives (TN) for the poacher class:

  $$\begin{aligned}\text{TN} = 1279275\,(\text{Passive correctly predicted}) + \\ 503\,(\text{Animal correctly predicted}) = 1279778\end{aligned}$$

Substituting these values:

$$\text{FPR} = \frac{155}{155 + 1279778} = \frac{155}{1279933} \approx 1.21 \cdot 10^{-4}$$

Each sensor generates one sample per second. The total number of samples generated by one sensor in a day is calculated as follows:

$$\text{Samples per sensor per day} = 1\,\text{sample/sec} \cdot 86,400\,\text{seconds/day} = 86,400\,\text{samples/day}$$

The expected number of false positives per sensor per day for the poacher class is given by:

$$\text{False Positives}_{\text{Poacher per sensor per day}} = \text{Samples per sensor per day} \cdot \text{FPR}_{\text{Poacher}}$$

Substituting the values:

$$\text{False Positives}_{\text{Poacher per sensor per day}} = 86,400 \cdot 1.21 \cdot 10^{-4} \approx 10.45$$

With 130 sensors deployed around the perimeter, the total number of false positives per day is calculated as:

$$\text{Total False Positives}_{\text{Poacher per day}} = \text{False Positives}_{\text{Poacher per sensor per day}} \cdot \text{Number of Sensors}$$

Substituting the values:

$$\text{Total False Positives}_{\text{Poacher per day}} = 10.45 \cdot 130 \approx 1358$$

This estimation suggests that the system could generate approximately 1,358 false positive alerts for the Poacher class per day, which is impractical for rangers to manage effectively.
To mitigate these high numbers of false positives, several strategies can be employed:
Rather than deploying sensors uniformly along the entire perimeter, sensors can be concentrated in critical or high-risk areas where poachers are known to enter frequently. By prioritizing these zones, the number of sensors—and thus the total false positives—can be significantly reduced.
For instance, if we reduce the number of sensors by half, deploying 65 sensors strategically, the expected total false positives per day become:

$$\text{Total False Positives}_{\text{Poacher per day with 65 sensors}} = 10.45 \cdot 65 \approx 679$$

While this is a reduction, the number of daily false positives remains substantial.
An effective way to further reduce the impact of false positives is to integrate accelerometers with fence-based camera systems. The accelerometers serve as initial triggers, activating cameras only when significant events are detected. This approach provides several benefits:
Benefits of this integration include:

- Enhanced Verification: Visual confirmation helps rangers quickly assess whether an alert corresponds to a genuine threat or a false alarm.
- Resource Efficiency: Triggered recording conserves power and storage by capturing data only when necessary.
- Improved Training Data: Over time, the visual data collected can be used to refine and enhance the accelerometer classifier's training, improving its accuracy.

By relying on visual verification, the system effectively reduces the operational false positive rate experienced by rangers. The combination of sensor data and visual confirmation allows for immediate assessment without the need for physical inspection. Moreover, integrating cameras can alleviate the necessity for the accelerometer system to distinguish between Animal and Poacher classes. The primary role of the accelerometers becomes detecting any significant fence disturbance Event vs. No Event, with the cameras providing the necessary information to differentiate between animal activity and potential poaching incidents.
This integration also addresses the challenge of labeling passive data without precise context. As discussed in Section 6.7, labeling passive data blindly can lead to misclassifications due to unknown events occurring during data collection. By having

visual records corresponding to accelerometer readings, we can improve the quality of our training data, enabling successive refinement of the classifier and potentially reducing the false positive rate over time.

7

Discussion

7.1 Model performance

Each model has achieved its own level of performance. In general, the problem formulations and system specifications are so distinct in each case that it is difficult to generalize the performance of all models.

The gunshot detection model appears apt in achieving a low amount of false positives - the primary goal of the thesis - while there is lots of room for improvement in both accurately detecting true positives and utilizing computationally cheaper features.

With the current dataset, no final conclusions can be drawn about the adequacy of decision trees for detecting elephant footsteps in a savanna environment. More data is needed, especially more diverse positive data. The combination of decision trees and a meta-model shows potential for creating an accurate and robust system, however, other models may be more accurate and provide an edge in performance that is required for the system to have both acceptable recall and false positive rate.

The fence intrusion detection model shows general promise, especially as only computationally cheap features are used. However, a more rigorous analysis of false positive sources between the classes is necessary to improve the model. Furthermore, since all case 3 data is re-enacted rather than originating from real poachers or animals, there are many sources of error to account for. Of course, for all cases, meta-models show promise of patching the holes of the initial models performance.

7.2 Discussion

Having had the opportunity to write such a interesting thesis has lead to many insights, both about the theoretical aspects of machine learning and hardware development, but also about what issues we are hoping to solve and what the reality

on the savannah is like. What follows is but a short summary of some of the insights we have gained.

During our time at Makulu, we found that collecting data was more difficult than anticipated. The presence of dangerous animals on the property meant that all excursions from the fenced-in lodge area had to be accompanied by experienced rangers. This generally entailed careful planning in advance since rangers tend to be hard at work from waking up to falling asleep. Passive data collection with remote systems was beneficial, as they only required the equipment to be swiftly installed and then left alone, and did not require our presence for active data generation or supervision. For active data generation, planning how to do it in advance was necessary, to be as efficient as possible when at the data collection site.

For case 2, several lessons were learned regarding hardware. We did not adequately anticipate the ability of elephants to smell gear that humans had touched, resulting in them breaking the geophone reading unit. In retrospect, this should have been considered, and more protective measures should have been taken. For example, chili powder is a known elephant deterrent, as their highly sensitive sense of smell does not tolerate the spice well. Furthermore, hardware development took much longer than we had thought. If this had been carried out more in advance, data collection in case 2 could have started earlier, spare hardware could have been constructed and in place, and potential problems could have been better assessed and addressed, resulting in more data of higher quality being collected.

One of the most significant lessons from this thesis work has been the hierarchy order in model creation, where in descending order of importance we find:

1. Data quality
2. Extracted features
3. Model
4. Hyperparameters

Each step down in the hierarchy is highly dependent on the preceding, so one should expect to spend more time on the higher hierarchy steps.

In the end, we hope that this thesis does not end up in a pile of research that doesn't contribute to the academic or wildlife community. We hope that, besides the technical details, we are able to shed some light on the pressing issues of wildlife conservation. What rangers have to say about the technical developments and potential are, in the end, equally if not more important than the technical nuances. When solving problems in bare environments, the simplest and most understandable solution is often the best. We hope that this thesis has underlined the need for bridging the

gap between the research community and the end-user. Often, lessons learnt on the savannah, such as the one from case 1 (I.1), are equally as valuable as those found in code.

7.3 Future Work

For future work, it is easy to point to further data collection. This is particularly true for elephant detection, where data was a limiting factor.
For gunshot detection and fence intrusion detection, more thorough analysis of what the results say, and what the sources of false positives and false negatives are, is recommended. Decision trees have shown themselves promising in maintaining low false positives rates in these simple detection problems. Further improving the potential of decision tree models lies in spending more time on careful analysis of each problem. As this thesis attempted to solve an array of problems in an unfamiliar environment, some shortcuts and trade-offs have been made - such as focusing on particular poacher behaviors for case 3. Taking case 3 as an example, an extension of the cascade classifier into several types of behavior, along with strong understanding of each type and it's associated features, combined with some simple rule-based system as a meta-model, could turn into a very potent fence intrusion detection system. For case 1, finding computationally cheaper features and further researching the reasons for false negatives stands out as fields of potential.
Furthermore, although the testing of alternative models (other than decision trees) had varied results in our thesis, case 2 experienced an increase in accuracy and decrease in false positives when experimenting with boosting methods. As such, there is still value in exploring alternative models. Models that are not based on features should not be excluded, as all our models require heavy-to-compute features at one stage or another, so there is not necessarily a gain in computation time from using a decision tree classifier. In case 1, it is found that feature computation constitutes 99.8% of the inference time, while the decision tree is only the remaining 0.2%. Furthermore, recall the layered approach of a top and a bottom classifier, as seen in case 3, where the top classifier weeds out the easy negatives, and a more intricate classifier distinguishes between the candidates that were not weeded out by the top classifier. Here, a very accurate bottom classifier could be feasible even if it is slow.

A

Code functionality in Case 2

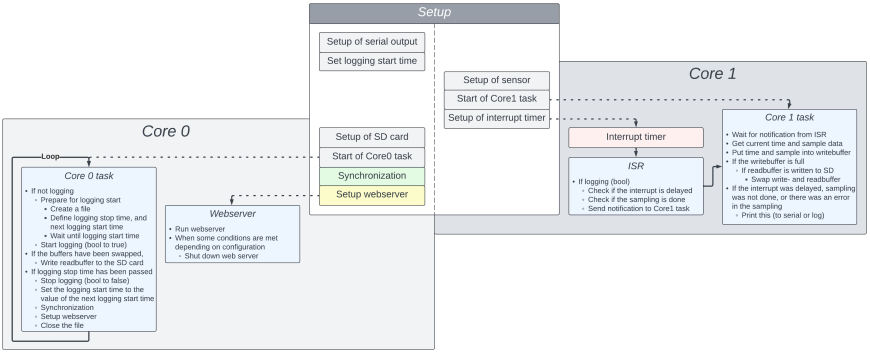


Figure A.1 The functionality of the code visualized. The web server is accessed by connecting to the WiFi access point of the camera and going to the camera's IP address in a web browser.

B

Feature correlation for Case 2

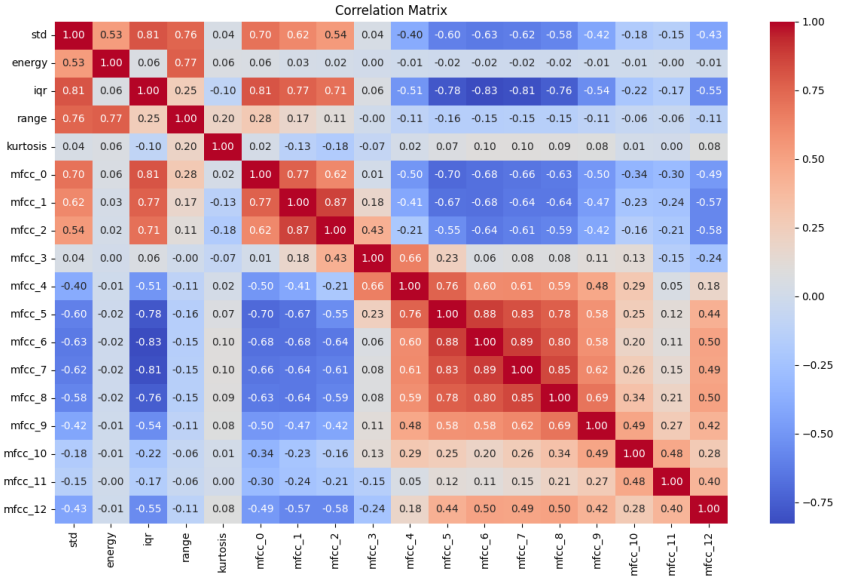


Figure B.1 Feature correlations before PCA.

C

Geophone and camera hardware problems

C.1 Geophone

The following problems were encountered with the geophone hardware setup:

1. ADS1256 not responding
2. Noisy output from the ADS1256
3. Partially broken geophone connections
4. The cables soldered to the 3.5 jack broke
5. Powering the LilyGo from the battery stopped working
6. The LilyGo unit was broken by elephants

The ADS1256 was originally not responding to commands, however, at a certain point it started responding, without any change of the setup. The output from the ADC unit had several types of noise, as seen in Figure 5.5. The algorithm used to

In going through the unit settings, attempting to fix this, it was found that when the analog input buffer was disabled, the signal was completely gone.The noises have been filtered out after collecting the data.

One of the geophones had partially broken cables at the connection to the geophone unit. These touched, creating noise. To fix this, without risking more damage to the unit, poster putty was used to isolate the connections. In the data collection, this geophone unit was not used.

The cables soldered to a headphone jack broke in the handling if the unit. Another 3.5 mm jack was used instead, but the cables on this were not long enough to place

the jack outside of the container. Instead, the jack was placed facing out of the container from the inside. To support it from side forces, a tube was glued around it.

The LilyGo unit did at one point stop taking power from the battery. Instead, a 10000 mAh power bank was placed inside of the container, and the unit was powered through the USB-C connector.

During data collection, a group of elephants dug up the container. They shook it around, breaking the components loose from the container. This caused the SD card slot on the LilyGo to break, making it unable to interface with SD cards. The currently plugged in SD card also broke. The event was captured by the camera, and is seen in Figure 5.4.

C.2 Camera

The following problems were encountered with the camera hardware setup:

1. LilyGo units breaking
2. SD card stopping responding
3. Cameras out of focus
4. Camera driver crashing with low-compression JPEGs
5. Solar panel batteries discharging quickly
6. Solar panel outputs broken

Two assembled camera units were collected, as well as two loose camera units. One of the camera units was broken, so another camera unit replaced it in the assembly. However, this unit was also soon found to be broken. It was discovered that the cables to the voltage regulator were incorrectly stripped, causing the positive and ground cables to touch. This caused large heat generation. It was assumed that this was the cause of the failure, so the stripping was redone, and another LilyGo unit was put into the assembly. Also this one broke after a while, seemingly leading to the conclusion that the cause of failure is the voltage regulator. The voltage regulator may have broken due to the heat generated.

One of the SD cards would randomly stop responding during use, and after that no longer be writeable until it was reinitialized. It was replaced, and the problem did not return.

The camera units were all more or less out of focus. The focus was adjusted by screwing in or out the lens. Some of the lenses were glued in place, but with enough force, the glue could be broken, and they could be readjusted.

The camera driver crashed when photos were taken of highly detailed scenes, with a low value of the compression parameter. To avoid this, a sufficiently high compression value was set.

Some of the solar panels did not hold their charge for more than a short time. This made it necessary to test every solar panel before using it. Some also had outputs that did not work.

D

Geophone datasets and their locations

Day	Camera	Length	Positive	Depth	Figures
6	No	1:07	0	10 cm	D.1a
7	No	1:02	0	15 cm	D.1b
8	Yes, 200°	6:42	0:20	20 cm	D.2a, D.2b, D.2c, D.2d
9	Yes, 200°	5:09	0	20 cm	D.2a, D.2b, D.2c, D.2d
10	Yes, 200°	7:19 (4:16)	0	20 cm	D.2a, D.2b, D.2c, D.2d
11	Yes, 200°	2:15 (1:02)	0	20 cm	D.2a, D.2b, D.2c, D.2d
12a	Yes, 200°	0:12	0	20 cm	D.2a, D.2b, D.2c, D.2d
12b	Yes, 200°	1:20	0	20 cm	D.2a, D.2b, D.2c, D.2d
13a	No	0:58	0	20 cm	D.2a, D.2b, D.2c, D.2d
13b	No	1:02	0	20 cm	D.2a, D.2b, D.2c, D.2d
13c	No	0:53	0	20 cm	D.2a, D.2b, D.2c, D.2d
13d	Yes, 200°	5:54 (3:25)	0	20 cm	D.2a, D.2b, D.2c, D.2d
14a	Yes, 200°	5:34	0	20 cm	D.2a, D.2b, D.2c, D.2d
14b	Yes, 200°	5:16 (4:33)	0	20 cm	D.2a, D.2b, D.2c, D.2d
15a	Yes, 200°	2:16	0	20 cm	D.2a, D.2b, D.2c, D.2d
15b	Yes, 200°	4:01	0	20 cm	D.2a, D.2b, D.2c, D.2d
16a	Yes, 200°	2:00	0	20 cm	D.2a, D.2b, D.2c, D.2d
16b	Yes, 200°	10:41 (7:17)	0	20 cm	D.2a, D.2b, D.2c, D.2d
19	Yes, 200°	6:31	0	30 cm	D.3a, D.3b, D.3c, D.3
21	Yes, 120°	6:00	0	30 cm	D.4a, D.3b, D.4b

Table D.1 The geophone datasets. All days are in November 2024. The times are in the format hh:mm. The data that can actually be used (due to the sun going down, or the camera being out of sync) is presented in paranthesis, if it is not the full dataset. The positive data time is where an elephant is present. The depth refers to how deep into the soil the bottom of the geophone was placed. In total, 62:20 of useable data are available. Since one sample is produced per second, this gives approximately 225000 samples for modeling and testing.

Note that the depth at which the geophone could be buried was limited, in the first two datasets, by the mountain rock below the soil. In the datasets where no camera supervision was available, the area around the geophone was visually supervised to ensure that no elephant was present. In the first two datasets, camera supervision is missing because the camera software was being developed, while in subsequent cases of no camera supervision, the software was being improved and debugged.

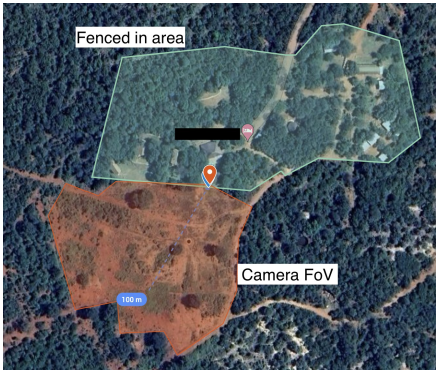


(a) The geophone placement on the 6th of November. ©2024 Bilder, ©2024 Airbus, Maxar Technologies.



(b) The geophone placement on the 7th of November. ©2024 Bilder, ©2024 Airbus, Maxar Technologies.

Figure D.1 Setups on the 6th and 7th of November, when data was collected without camera supervision.



(a) The geophone and camera placements on the 8th to 16th of November. ©2024 Bilder, ©2024 Airbus, Maxar Technologies.



(b) The geophone unit, the way it was placed on the 8th to 16th of November in sunny weather.

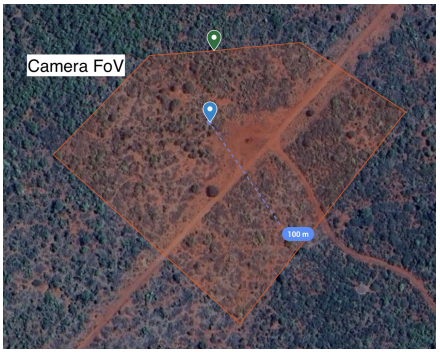


(c) The camera, the way it was placed on the 8th to 16th of November in sunny weather.



(d) The camera, the way it was placed on the 8th to 16th of November in rainy weather.

Figure D.2 Setup from the 8th to 16th November, when the majority of the data was collected.



(a) The geophone and camera placements on the 19th of November. ©2024 Bilder, ©2024 Airbus, Maxar Technologies.



(b) The geophone unit, the way it was placed on the 19th and 21st of November.

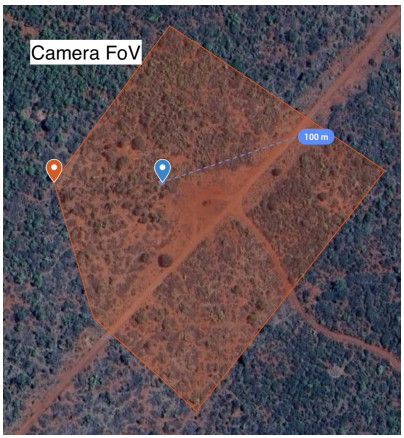


(c) The camera, the way it was placed on the 19th of November.



(d) The setup on the 19th of November.

Figure D.3 Setup on 19th November, the first dataset away from the lodge area. Note that the marked square in Figure D.3a is an estimate of an open area which was cleared after this aerial photo was taken.



(a) The geophone and camera placements on the 21th of November. ©2024 Bilder, ©2024 Airbus, Maxar Technologies.



(b) The camera, the way it was placed on the 21th of November.

Figure D.4 Setup on 21th November. The geophone placement is found in Figure D.3b.

E

Reconstruction of elephant footstep PSD from MFCCs

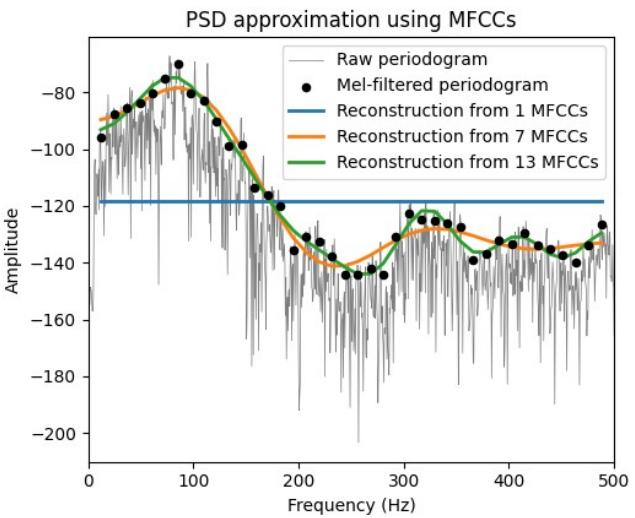


Figure E.1 Figure demonstrating how the MFCCs are a compact way of representing a PSD by reconstructing it from their values.

F

Full interviews

F.1 Gunshot Detection

Tiaan A.

Tiaan A. has worked on a different, large, farm that utlilizes shotspotters. His experience and feedback provides insight into many factors to consider when designing a shotspotting system. When asked what shortcomings he found in the shotspotting system he had seen before, he highlighted two major areas: false positives and maintenance. As a farm manager and ranger, he emphasized that time is precious. He points out that loud noises like an elephant breaking a tree when eating can trigger the shotspotter, leading to time and manpower being wasted to address a false positive. He notes that a system where audio information is sent to a ranger in the case of detection can be used to add human assessment and reduce wasting resources. Furthermore, the system he encountered made use of many individual shotspotters, causing a lot of time and manpower being spent on maintenance. Not only does a knowledgeable person that knows the hardware need to be available, but working on the savannah means they also need to be accompanied by at least one more person for their safety when it comes to wild animals.

When it comes to what Tiaan would like to see in a shotspotting system, Tiaan explained that it would be very useful to be able to distinguish between different calibres of shots fired. A higher calibre rifle indicates luxury poachers that are after large game, such as rhino and elephant. Lower calibre rifles indicate meat poachers that are looking for animals such as Kudu, Impala and Zebra. Having such information at hand could help rangers determine the level of action needed and the seriousness of the threat.

When it comes to the placement of shotspotting sensors, Tiaan suggests a concentration around the territories of targeted animals, such as rhinos. However he also points out that shotspotters placed outside of hearing range for the rangers is especially crucial.

Tiaan was also able to provide information about the strategies used by poachers, useful for the general development of shotspotting systems:

- Luxury poachers and meat poachers are generally active during different times of the day. Meat poachers tend to poach during the day, around lunchtime. This is when the farmer is having food and taking a break from the heat. Luxury poachers on the other hand often operate at night, between midnight and sunrise.
- Luxury poachers tend to be more active before holidays, and at the end of the month, when the need for money is greater.
- Luxury poaching teams are aware of the technology that is used for anti-poaching, highlighting the need for discretion when placing and designing the hardware.
- Both luxury and meat poachers are known to use of make-shift rifles and suppressors, making detection a potentially harder problem.

Didi S.

Didi Schoeman is the Chairman and Managing Director of Makulu. As such, he provides a wide perspective on the potential development of systems of sensors at Makulu. His background in business, accompanied by an MBA, further brings a long-term perspective of the sensor systems and their potential as a revenue-source.

In regards to shotspotting, Didi highlights that poachers most of the time use silencers, making detection a harder problem. He notes that systems with high detection range and localization capabilities would be useful in order to state if the shot is coming from within the property or from a neighbouring property. However Didi also believes in preventative measures, which he points out that shot-detection is not, only providing information after the fact that a gun has been shot.

Didi takes an approach that prioritizes multi-purpose systems. He for example explains that microphones able to detect not only gunshots but also drones would be very useful. Poachers need to scout the property before entering in order to find the animals of interest, often rhinos, and have been known to do so with the use technological solutions such as drones. With localization capabilities, it would also enable the ranger to track from where the drone has been flown and roughly where the user most likely is. He does say that, in case gunshots can be recorded, they could potentially be used in prosecuting poachers and gathering information for that purpose.

When asked about the range and placement about shotspotters, he notes that each farm is in of itself a case study, due to the environment and geometry being vastly

different from case to case. With mountains or hilly areas, he explains that localization algorithms can struggle with echo. If he had to generalize, he would say that most farms in South Africa are flat. But other countries, such as Mozambique, contain many hilly areas and the problem is different in its nature. He reasons that one of the most important things in the development of microphone systems is their adaptability. A system should be adaptable and very easily tunable by a ranger such that it works optimally for the terrain it is placed in.

Didi points out the need for easy-of-use. Having a system that is easy to understand and use is crucial for it being beneficial to the ranger. He also says that the set-up and maintenance should preferably happen off-site, needing no person on-site to tune sensitive or hard to understand settings. When it comes to communication between the sensor and the ranger, he would like an app-based system that is able to tell you the certainty of the detection.
Didi explains that the battle against poachers is constantly evolving, requiring thought through placement and camouflage. In the same way, systems should become better and better over time, learning in real-time.

He finally points out that false alarms are important to reduce, and that costs are important to keep low.

Junel D.

Junel Dry currently works at Makulu as Volunteer Coordinator and has experience in animal welfare and animal behaviour. She is involved in several of the technical advancements happening at Makulu and provides the perspective of a tech-savvy ranger.

Junel, like Tiaan A, has previously worked at a large farm in South Africa. When asked about her previous experience working with shotspotter systems she explains that while she herself has not directly worked with the systems, she knows a bit about how they've been utilized. Shotspotter systems are generally placed on top of koppies (the South African term for hills in otherwise generally flat areas) in order to increase range and lessen the risk of sound being blocked by the koppie itself. On top of the koppies, shotspotter systems are mounted on man-made poles, increasing height even further. Junel notes that the systems she has had contact with however required heavy equipment for maintenance, and their placement therefore implied using helicopters to transport such equipment.

Similarly to Didi, Junel reasons that shotspotter systems in of themselves are a

retroactive measure. They do not actively protect the animals, but rather gather evidence that potentially could be used for prosecution. When asked what she would wish from a shotspotting system, she responds with multipurposing the sensor. If a microphone-based system could be used to detect the loading of guns, that would be more of a preventative than retroactive measure. Poachers always use a drop-off and pickup system, using one or two vehicles to arrive to and leave from the target site. Having a system that could detect the sound of arriving vehicles, or perhaps the sound made when vehicles stand on idle, would increase the use-cases and impact of microphone systems as preventative systems.

Junel has several thoughts about the workings of a microphone-based system, raising questions regarding how to avoid false detections during thunderstorms and lightning. She further explains in detail the different types of sounds that can occur in a poaching situation. First, guns emit what is called in South African slang a "knappenknall" sound. The sound of the bullet leaving the muzzle (knapp) is followed by the sound of the bullet breaking the sound barrier (knall). A potential third sound that can be picked up by the human ear is the impact of the bullet hitting a target. Junel explains that rhino-poachers often aim for the shoulder or skull area in order to incapacitate the animal, and that the impact of a bullet on hard bone makes a distinctive sound that can be heard for up to hundreds of meters.

Junel further highlights several factors that motivate making a model highly generalizable. She notes that poachers often use makeshift guns, estimating that this is only slightly more common among meat poachers than luxury poachers. The makeshift guns she has encountered typically use shotgun cartridges loaded with three smaller-caliber bullets, which often severely wound the animal rather than kill it outright. She also points out that bullets can pass through or graze trees on their way to the target. This increases the possible sounds to detect but requires a generalizable model. Junel brings up the scenario of elephants breaking branches, explaining that while the initial sound of an elephant snapping a branch can resemble a gunshot, it is always followed by a ripping sound as the bark is torn from the tree. This suggests that a robust model could be developed but would likely require both additional training data and more complex models that incorporate sequential time information, such as LSTMs.

As a final note, Junel believes that the more information the better, even with the risk of crowding the rangers with notifications, as long as a human verification layer is available. She also highlights the Meerkat Wide Area Surveillance Systems

(WASS), used in several parks in South Africa, as a source of inspiration.

Lucas G.

Lucas G. has more than a decade of on-site experience in anti-poaching and farm management. Although he has never personally worked with shot-spotting systems, he is familiar with a neighboring sanctuary that employs them.

One key issue Lucas identifies is the high rate of false alarms reported by the neighboring sanctuary's shot-spotting system. False alarms not only waste the time of the system's owners but also disrupt neighboring farms and sanctuaries that cooperate in anti-poaching efforts. According to Lucas, a recurring source of such false alarms is the sound of elephants breaking tree branches.

Lucas stresses that if multiple sensors are used, low maintenance should be a top priority. Managing extensive maintenance tasks can take time from other equally critical responsibilities. He highlights battery life as a crucial factor: equipment failures often come from battery issues, and the need to replace batteries in animal tags, for instance, might require sedating and darting the animals—an outcome that should be minimized.

When asked about the strategies used by poachers when it comes to weapons, Lucas explains first and foremost that rhino poachers will use large calibre rifle such as .375 and .458. Most poachers employ some sort of silencer, which can be both homemade or purchased. Lucas estimates that a large majority of silencers he has found during his anti-poaching work has been homemade silencers. Therefore, it is important for any shotspotting system to be able to identify both silenced and unsilenced shots, regardless of silencer type. Lucas also explains that while a single shot generally is enough to achieve a poacher's goal, if more than one shot is taken they will always follow in quick succession - within seconds. This could potentially be used to aid a shotspotting system.

When asked about potential problems faced when developing shotspotting systems Lucas reiterates maintaining low false positives. Furthermore, an equally different problem is found in that Makulu has many neighboring farms where legitimate hunting activities take place. Lucas underscores that shot-spotting systems must reliably minimize false positives and aid in distinguishing legitimate hunting from poaching.

In terms of deployment, Lucas suggests placing shot-spotting systems near rhino habitats and other high-risk areas. He also suggests incorporating contextual knowledge, such as the times of day when poachers are most active (e.g., from after midday until around midnight), to enhance the system's effectiveness. Additionally, maintaining strict confidentiality regarding sensor placement and workings is essential to prevent information leakage.

Finally, Lucas states that attaining direction and location would be extremely helpful in their anti-poaching efforts, for catching the poachers after the crime has been committed. If the shotspotter could distinguish between different calibers, knowing that poachers generally use different guns from those use by farmers, could aid in distinguishing between legal and illegal shots.

F.2 Footstep detection

Tiaan A.

Tiaan has never worked with footstep detection systems before.
When asked about what a geophone system should be able to do, he explains that it would present an extra source of information. Several important animals on Makulu have tracking collars, however these ping with set intervals, and are sometimes not reliable due to battery failure or other problems. As such, a geophone system could aid in pinpointing the whereabouts of important species such as rhino and elephant, and giving the rangers a new source of information. He explains that all animals have favourite spots, and that figuring these out is a constant job of a ranger. Geophone systems could potentially help rangers know where to place extra sensors such as cameras, and where to prioritize security resources.
Furthermore, the potential to detect human footsteps is very valuable. Combining a geophone system with meta knowledge, such as when it is reasonable for rangers or volunteers to walk around, could provide a valuable warning system for poachers entering the property. If the system can estimate pace and direction, setting up ambushes for the poachers could become much easier.
When asked about knowledge that might help building a robust system that can account for known patterns of behavior, Tiaan highlights patterns of behaviour for both poachers and animals. He explains that while most rhino poachers wear normal shoes, some poachers have been known to be barefoot, wear only socks or even place diapers or hoofs under their shoes - all in order to make detection and tracking a harder problem for the rangers. Elephants and rhinos also have particular behaviours when it comes to walking the grounds. Elephants tend to drag their feet while rhinos will often walk with their nose touching the ground. Finally, rhinos are known to kick the ground after urination, in order to make the scent stick onto their

feet and thereby mark their territory while walking around.
When asked about potential problems one might encounter when building a geophone system, Tiaan thinks that it is important to avoid false alarms created by other animals and vehicles passing by.
For the placement of geophone systems, he explains that he would place them along the borders of the farm to detect poachers. He would also prioritize routes where animals are known to walk.
When asked about particular wishes for the system, Tiaan explains that he sees the system as a potential anti-poaching tool. Therefore he would narrow the scope of detection down to certain cases of interest: detecting rhinos, buffalos, elephants and humans. If the system would be able to estimate pace and direction that would be very useful.

Junel D.

Junel has not worked with footstep detection systems before.

She explains that the most important part of such as system would be to distinguish between human, animal, and mechanical vehicles. Similar to Tiaan, she reasons that tracking and knowing the movement of important species such as rhino and elephant is very useful. Knowing such things can inform the placement of other sensors such as cameras, and build a holistic system where different sensors work together.

When asked about strategies used by poachers and movement patterns of animals that could be good to know about when designing footstep detection systems, Junel explains that:

- Poachers use hoofs, diapers or socks on their shoes to increase stealth and make them harder to track.
- Elephants drag their feet and will kick up dust and move their trunk on the ground even when standing still and bored. Elephants also rumble.
- Rhinos eat soil and drag their mouth on the ground continously.

When asked about potential issues to predict, Junel explains that electronics are always targeted by elephants as they target the scent of humans. Furthermore the devices need to be water resistant, insect resistant and have good insulation overall.

For the placement of the geophone module, Junel explains that she would place it in animal hotspots such as watering holes, feeding spots and mud wallows, as long

as it is well-hidden.

Finally, Junel says that one interesting area of application of such a sensor could be to detect when mating rituals occur. When rhino bulls fight, it indicates that something is potentially happening in the herd hierarchy. Knowing when there is a high chance of important species such as elephant and rhino having a baby would prove useful for planning relocation, resource management etc.

Didi S.

Didi has not worked with footstep detection systems before, but has read about using fiber optics for footstep detection. The research he found had a directional detection area, which was very interesting for protecting the fence line. Such systems are however just too expensive, he recalls a price point of roughly 300 000 rand per kilometer. With this, Didi once again raises the question of affordability.

The most important part of a footstep detection system for Didi is achieving a low false alarm rate and distinguishing between people and animals. He explains that poachers often strap diapers and hoofs on their feet, making the problem a lot harder. He recommends looking at the tempo and gait of any detection. Furthermore, one can use the sensor to detect when both animals and poachers are digging holes near to the fence, which provides vital information in both cases.

When asked about strategies used by poachers or animals that could be good to know about, Didi explains that meat poachers often come with packs of dogs, which might be easier to pick up or give another source of information.

For the placement of the sensor, Didi reasons that a circular detection radius makes the sensor suboptimal for placement close to fence (where a directional sensor might work better). He in this case would rather place it in animal hotspots or high risk areas inside the perimeter.

Finally, Didi explains that another purpose for footstep detection systems could be to protect trees. When elephant footsteps are detected nearby an endangered or important tree, a defense system (such as pepper spray or the sound of bees playing from a speaker) could be activated to deter the elephant from knocking the tree over.

Lucas G.

Lucas has not worked with geophone systems before.

After poaching, poachers will always run for 100-200 meters from the site of the crime. From there on they will walk, potentially even stop and put their heavy bags down, and then continue walking. Behavior that can potentially be detected using geophone systems.

Poachers will use socks, and also walk on elephant and giraffe tracks. This makes their footprints much harder to detect. Therefore Lucas suggests developing footstep detection in this order: elephant, giraffe and human.

When asked about potential problems faced when developing footstep detection systems, Lucas highlights elephants potentially damaging the system, if they can smell human scents. He also highlight soil erosion potentially affecting the sensors placement environment.

Lucas would place the sensor in animal and poaching hotspots.

Finally, Lucas stresses the importance of technology development. He explains clearly that sanctuaries that employ less technological tools to combat poaching, experience much larger amounts of poaching. He stresses that technology constantly needs to improve and develop as the poachers are doing the same. It is a never ending battle and he says to think long term rather than short term. Develop systems that can be developed even more down the line.

F.3 Fence intrusion detection

Junel D.

Junel Dry, the Volunteer Coordinator at Makulu, brings valuable insights from her background in animal welfare and behavior, coupled with an interest in leveraging technology for conservation efforts. Although she has not worked directly with fence detection systems, she is familiar with their application in large game reserves across South Africa. She points out that fence detection systems differ fundamen-

tally from shot-detection systems, as they aim to prevent incidents by identifying poachers before any harm comes to the animals.
Junel is able to identify several strategies used by poachers to both breach fences and cover their tracks:

1. Cutting the fence entirely to gain entry.
2. Placing hollow plastic tubes around the wires to avoid electric shocks, and bending the wires apart to squeeze through.
3. Cutting binding wires on poles to facilitate bending the wires.
4. Short-circuiting the fence by grounding wires at multiple points.
5. Using quari bush to obscure tracks, making it appear as though only birds have been present.
6. Wearing socks or diapers over shoes to hide footprints.
7. Testing wires with sticks to check if they are electrified.
8. Entering through warthog holes under the fence, if they are large enough.

She also highlights several challenges in implementing fence detection systems:

- Weather conditions: Wind and rain can interfere with the system's performance.
- Animal interference: Large animals, such as kudus jumping over or into fences, can damage or dislodge sensors. However, she notes that elephants generally avoid fences, and cameras placed on fences in the past were left undisturbed by them.
- Camouflage and sturdiness: Sensors should be difficult for poachers to notice and hard to remove once discovered

Junel identifies additional issues that currently affect the effectiveness of fences:

- Grass and overhanging branches frequently cause the electric fence to short-circuit.
- Battery failures are common, reducing the fence's functionality.
- Plastic poles used in the fence can melt during nearby bushfires (see 6.4).

Finally, she sees additional uses for fence detection systems beyond anti-poaching measures. These systems could monitor animal enclosures to detect when animals attempt to escape or when predators try to enter. However, such systems would need to be durable and controllable, especially since the fence of enclosures are often touched during feeding and maintenance of enclosures.

Tiaan A.

Tiaan A., a ranger at Makulu, has not previously worked with fence detection systems nor directly interacted with them. However, he is aware of a warning system that was once implemented on Makulu's fence. This system involved installing short-circuit detection mechanisms in the power boxes supplying electricity to the fence. When a short circuit occurred—such as when two wires were touched together—a connected speaker would sound an alarm, and notifications would be sent to rangers. Unfortunately, this system is no longer functional.
When discussing important features of a fence detection system, Tiaan emphasizes functionality and reliability as the highest priorities. Currently, rangers patrol the fence several days per week to ensure its integrity and check for damage that may have occurred overnight. A system capable of reducing this workload would provide significant value.
He explains that elephants frequently push trees onto the fence, causing extensive damage that leaves large sections vulnerable to breaches. Such damage poses risks not only from poachers but also from animals escaping into neighboring properties. He highlights the serious consequences of these escapes:

- Neighboring game hunting ranches have the legal right to hunt escaped animals within a few days unless the animals can be recovered and are tagged.
- Escaped elephants can also be legally classified as "problem animals," making them targets for hunting.
- Such hunts are lucrative opportunities for hunters, creating an additional economic incentive to target escaped wildlife.

Tiaan identifies several challenges that a fence detection system might face, particularly related to animal interactions with the fence:

- Animal collisions: While larger animals like kudus occasionally jump over and get stuck in the fence, he explains that smaller animals are more commonly involved. These incidents occur more frequently, involve lower-impact collisions, and tend to happen closer to the ground.
- Poacher tactics: Poachers employ various methods to breach fences:
    1. Entering through warthog holes if they are large enough.
    2. Cutting the wires.
    3. Opening the fence and squeezing through the gap.
    4. Climbing over the fence if the electricity is off.

> Poachers often carry electrical testers to determine whether the fence is powered, allowing them to act accordingly.

Finally, Tiaan emphasizes that sensors should be discreet and easy to conceal. Ideally, they could be hidden inside the plastic tubes already present on Makulu's perimeter fence (see section 6.4). Additionally, he believes the sensors should contribute to maintaining the fence in a fully operational state, minimizing downtime and ensuring continued effectiveness.

Didi S.

Didi S., the Chairman and Managing Director at Makulu, has been part in initiating the exploration of fence-detection systems. The idea arose when poachers began using hollow pipes to cover the electric fence and climb through it. This tactic allowed them to bypass Makulu's earlier fence protection measures, highlighting the need for a new and more effective solution.
When discussing the most valuable aspects of a fence-detection system, Didi emphasizes the importance of detecting different types of poacher activity, such as climbing over, climbing through, and cutting the wires. Additionally, being able to accurately locate where the activity is taking place, within a reasonable margin of error, and minimizing false positives are crucial factors.
Regarding strategies used by poachers to breach fences, he notes that going undetected is of utmost importance to them. Poachers have been known to place diapers, hoofs, and socks over their shoes to avoid leaving footprints. Therefore, concealing the detection system is vital. If a system is visible, poachers will eventually find ways to bypass or destroy it. Didi suggests placing the sensors inside the fence poles or separating the communication module from the analog sensor. By separating these components, the module can be hidden inside the property and connected via a concealed wire to the sensor part.
When addressing potential challenges in developing fence-detection systems, Didi highlights several issues:

1. Lightning Strikes: Lightning can travel through the wires and fence, potentially damaging electronic components.

2. Insect Infestation: Ants and spiders may nest in or accumulate on electronics, causing malfunctions.

3. Environmental Exposure: Devices need to be insect, rain, and dust-proof to function effectively in the bush.

Regarding the placement of fence-detection systems, Didi recommends installing the sensors inside fence poles to prioritize low visibility and protect them from environmental factors.

Finally, he underscores that affordability is the primary reason systems like these fail. An affordable system with good precision and the ability to pinpoint where an event is happening is key. Didi advises future researchers to focus on communication systems—whether that be Wi-Fi, Bluetooth, cellular, or radio. The expense of data transmission and maintenance, rather than the initial cost, is often the bottleneck for anti-poaching systems.

Lucas G.

While Lucas G. has not directly worked with fence detection systems, he is familiar with devices that notify rangers when a short circuit occurs on the fence. This experience provides him with some insights into how to leverage existing fence infrastructure.

Lucas emphasizes that any fence detection system must remain hidden from poachers. He suggests integrating sensors into various fence elements—such as different types of poles—so that the devices remain hidden in the existing structure. Additionally, accurate localization of detections is crucial. Given the extensive perimeter of Makulu, pinpointing the location of a breach can save time and resources. Incorporating contextual knowledge, such as during what hours of the day there is a heightened risk of poacher activity, can further enhance the system's effectiveness.

When asked about the behavior of poachers when it comes to fence intrusion, Lucas was able to highlight two new ways that he has observed poachers using:

- Utilizing nearby trees to climb over the fence.
- Making use of wooden ladders to climb over the fence.

He does note that both these behaviors he has only observed once. They are therefore much more rare than those mentioned by Tiaan in F.3 for example, which Lucas also explains are the most common strategies.

When considering potential issues in fence detection development, Lucas stresses the importance of building sensors that withstand rain and weather conditions. Sensors should be as small and discrete as possible. Furthermore, the system must be resilient against short circuits or other electrical anomalies along the fence, ensuring that these disruptions do not affect sensor functionality.

Lucas explains that while Makulu's fence spans approximately 26 kilometers, not all segments are equally vulnerable. He estimates that by covering around two kilometers, it would be possible to monitor all known hotspots where poachers are most likely to breach the perimeter.

When asked about future developments, Lucas says that the development of drones and sensor networks could potentially be game changing for anti-poaching efforts. For instance, if sensors could communicate in real time and automatically dispatch autonomous drones to intrusion sites, anti-poaching efforts could become much more effective.

G

Software Pipeline

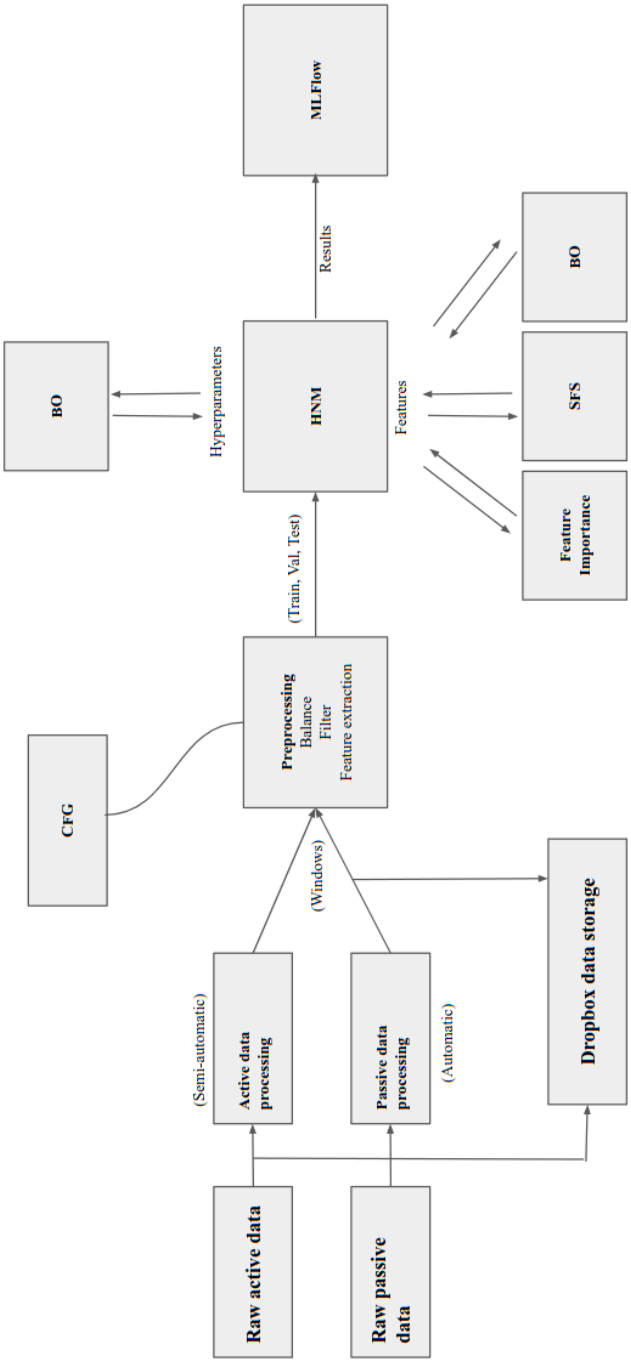


Figure G.1 Overview of the software pipeline in broad strokes.

H

Parallel projects

H.1 Introduction

Makulu Sanctuary hosts a large amount of sensors and technology that it uses for a multitude of purposes. This stems from the general belief in that technology can solve or aid a lot of the problems that exist on the savannah, as well as from a cooperation with Linköping University. However, being located on the south african savannah and having a budget that is strained between many different areas of need, Makulu most of the time does not have engineers on-site. So when the, often newly-developed, technology fails, trouble-shooting the problem can become very hard. As students on-site for almost three months, we provided some valuable expertise even outside of our areas of focus.

H.2 Weather station

Spearheaded by Carlos Vidal, Makulu Sanctuary contains a weather station that collects and monitors environmental parameters such as wind speed, rainfall and humidity. About one week after the previous developer team left Makulu, the weather station stopped sending data.

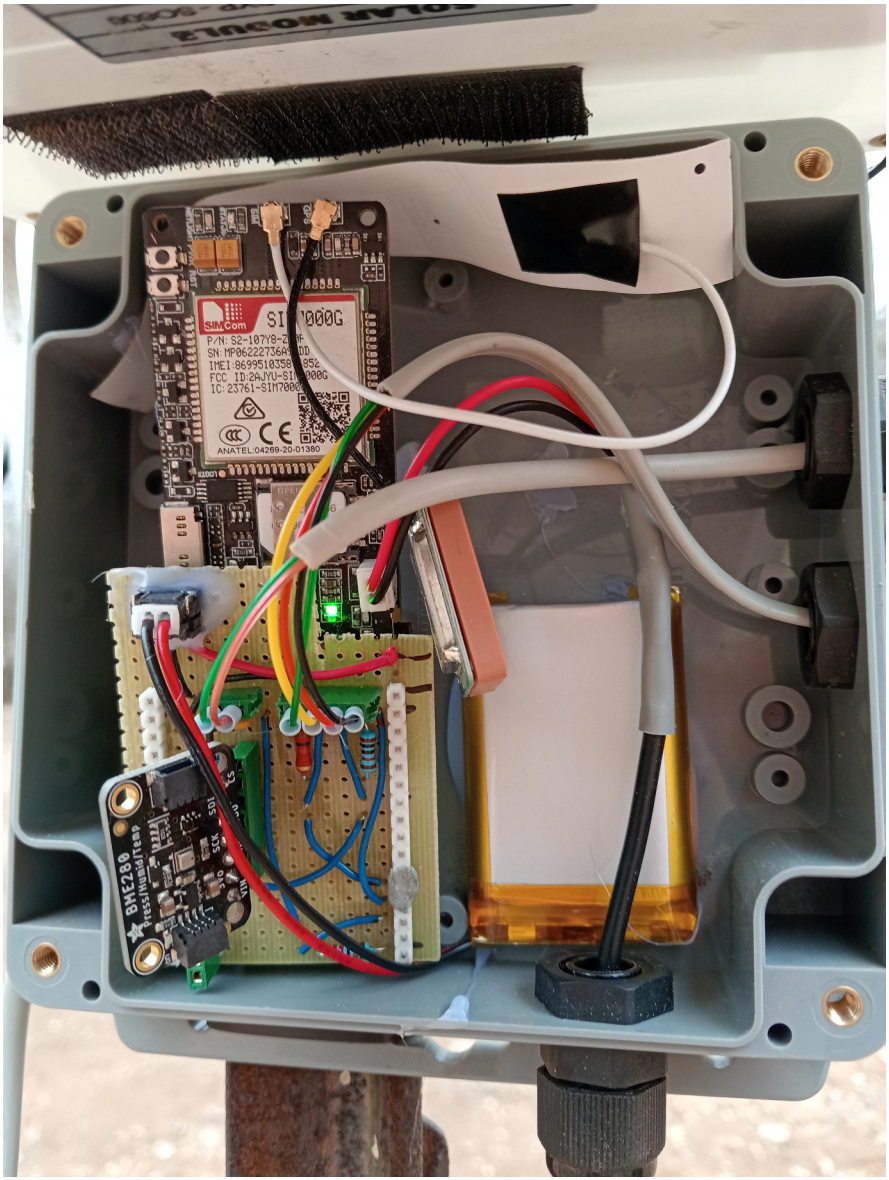


Figure H.1 Weather Station Setup at Makulu

The weather station had first of all had it's external cables bit through by an animal (monkeys most likely). This needed to be fixed by resoldering the wires color by color and reattaching. Furthermore, the sensor was not sending data as it should

even without cable connection to the sensors. Several attempts were made at fixing this issue by flashing the board with new firmware. In the end, it was decided to bring the main board of the weather station back to Sweden so that a new solution could be implemented.

H.3 Scanners

Spearheaded by Mikael Fredriksson, Makulu Sanctuary uses about a dozen small scanner stations. The scanner stations are placed around the sanctuary and activate by bluetooth when an animal collared with an activity-tag approaches. When pairing and connection is established, a datavector is passed from the activity-tag to the scanner and from there to the Qulinda system. Some scanners were communicated to be non-functional, as they were not sending data.



Figure H.2 Scanner Station Setup at Makulu

Upon investigation, it was realized that no scanners were functional in the field. For some scanners, their partition system was not compatible with the software. Several attempts were made at flashing new firmware on to the scanners, with temporary

success. In the end, it was decided to bring certain scanner modules back to Sweden, for an upgrade.

H.4 Lodge fence electricity

The lodge area at Makalu was surrounded by an electric fence, which stopped working at some point. This enabled an elephant to break through the fence, which it did every night for a week, eating from the stockpile of grass, knocking down trees, and causing a safety risk. The broken part was a contact in an energizer. We resoldered this contact, after which the energizer was put back in use, stopping subsequent attempts from the elephant to break inside.

I

Additional results

The following chapter is used to show additional and relevant results that were obtained during the project development.

I.1 Gunshot detection

Wind noise

The results of the model were qualitatively judged by, amongst other things, listening to the misclassifications made by the model. In this way, it was identified that noise impulses caused by wind led to the model predicting false positives. False positives, as highlighted in section 4.3, are a major concern in the development of shotspotters. Therefore, several efforts were made to reduce the false positives caused by wind noise. Below we present three attempts at combating wind noise and the results of each. Note that the confusion matrices presented here were gathered in an early stage of the project implementation. The baseline results before testing the three strategies is presented first:

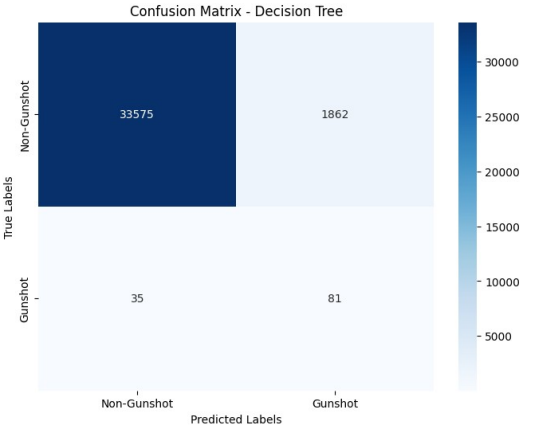


Figure I.1 Confusion matrix for gunshot detection at early project development stage, before combating wind noise



Figure I.2 Hardware adaptation to reduce wind noise impact - using a simple cloth

Hardware adaptation. Figure I.2 shows the hardware modification tested for simple wind noise reduction.



Figure I.3 Confusion matrix for gunshot detection at early project development stage, testing to collect passive data with a cloth adaptation

Figure I.3 indicates that when testing the system on data collected with a cloth over the microphone, the number of false positives was significantly reduced.

Wavelet transform. The wavelet transform is a signal processing tool that allows decomposition of a signal into different frequency bands, providing both time and frequency localization. Unlike the Fourier transform, which relies on infinite sinusoidal functions, the wavelet transform uses short waveforms called wavelets to analyze localized variations within the signal, making it effective for transient signal detection, such as gunshots.
The continuous wavelet transform (CWT) of a signal $x(t)$ is defined as:

$$W_x(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t) \psi^* \left(\frac{t-b}{a} \right) dt,$$

where:

- $\psi(t)$ is the wavelet function (also known as the mother wavelet),
- a is the scale (related to the frequency content),
- b is the translation (related to time shifts), and
- ψ^* denotes the complex conjugate of the wavelet function.

To denoise a signal, the wavelet transform can be applied to decompose it into multiple scales. By analyzing and thresholding the wavelet coefficients, low-frequency

components associated with noise (e.g., wind) can be suppressed while preserving high-frequency components related to sudden events like gunshots. This process enhances the signal clarity, making it easier to distinguish between true gunshots and background noise. Thus, wavelet transform preprocessing can help reduce false positives in noise-prone environments, while keeping the essential features of the signal intact.

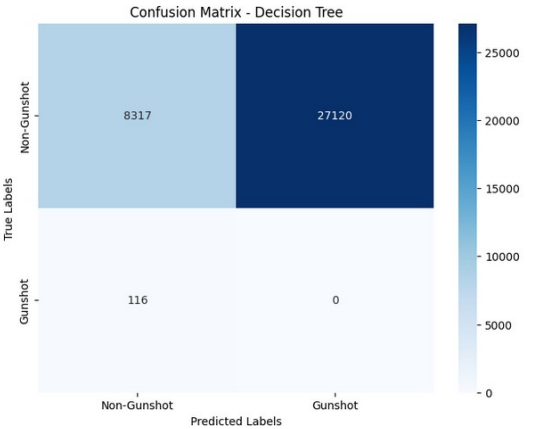


Figure I.4 Confusion matrix for gunshot detection at early project development stage, applying wavelet transform on all data in the preprocessing step of the pipeline. Transform was done with db4 and level 4.

Figure I.4 suggests that the wavelet transform has suppressed the gunshot features to the extent that they are indistinguishable from non-gunshot events, which might have caused the model to fail in detecting actual gunshots.

Highpass Butterworth filter. To preprocess sound files for gunshot detection, a high-pass Butterworth filter can be applied to remove low-frequency noise and enhance the sharpness of gunshot signals. The Butterworth filter is ideal for this purpose because of its smooth frequency response and minimal phase distortion. By setting an appropriate cutoff frequency, the filter suppresses lower-frequency components typically associated with ambient noise, such as wind or distant traffic, while preserving higher-frequency components.

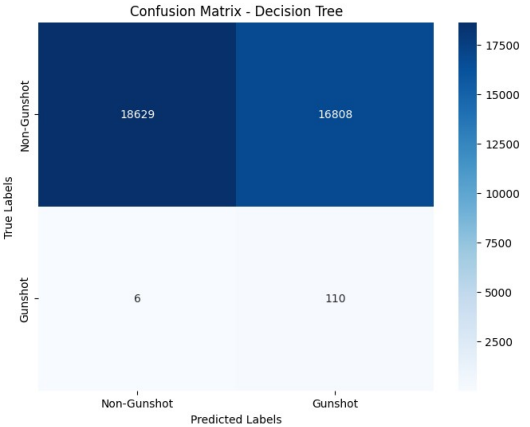


Figure I.5 Confusion matrix for gunshot detection at early project development stage, applying highpass Butterworth filter on all data in the preprocessing step of the pipeline.

Figure I.5 highlights a significant number of false positives, even after the application of the highpass Butterworth filter. This might suggest that, while the bandpass filter was intended to reduce noise, in this case, it may have had the opposite effect by skewing the noise profile of all samples towards gunshot-like characteristics, amplifying the problem. Instead of improving detection accuracy, the filter seems to have worsened the results significantly. This highlights the need for more precise filtering techniques or a tailored approach to handling wind noise.

SFS landing in local optima

When experimenting with feature selection, it was noted that, depending on the features and the test data, the optimized scoring function could be improved by optimizing for another scoring function. For example, False Positive Rate (FPR) could sometimes improve by optimizing for accuracy rather than FPR itself. This was hypothesized to occur due to SFS being a greedy algorithm that can get stuck in local optima. Unless the features space is small, SFS can be insufficient for providing a trustworthy set of features and results. The possible solutions to this problem lie in many places. For example, creating a more robust test and training dataset, where there is a larger overlap between dataset quality and characteristics between test and train, could help the model to not have such a widespread set of local optima. Furthermore, manually removing unimportant features could narrow down the feature space and reduce the risk of non-global optima. Thirdly, an alternative features selection algorithm that provides more global optimization, such as Bayesian optimization, could provide a more robust and trustworthy result.

I.2 Fence detection

Additional results

Top classifier. For the top classifier, using all features attains the following results:

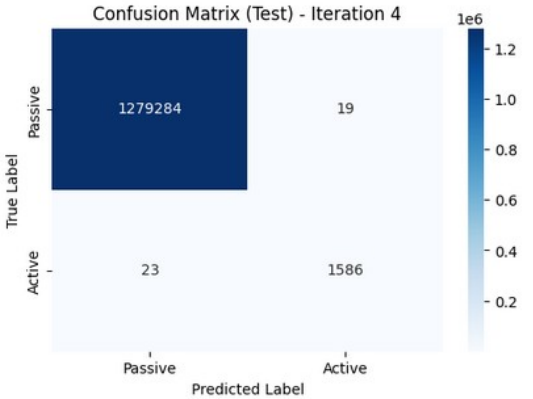


Figure I.6 Confusion matrix for fence intrusion detection top classifier using all features with HNM for four iterations

Metric	Value
Accuracy	1.0000
Precision	0.9882
Recall	0.9857
F1-Score	0.9869

Table I.1 Quantitative metrics for fence intrusion detection top classifier using all features with HNM for four iterations

For the top classifier, using feature importance as a feature selection method resulted in the following:

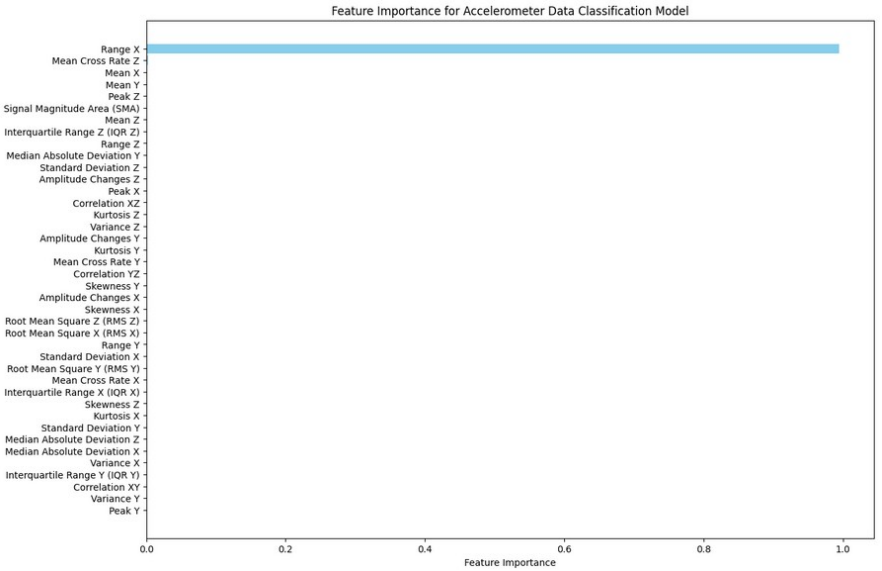


Figure I.7 Plot of feature importance from first classifier, highlighting that one feature dominates the decision-making capabilities to an extreme extent.

By cutting of features with less than 0.0001 importance, in other words features that contribute to less that 0.01 % of the decision making capabilities, the features left become:

1. Range X
2. Mean Cross Rate Z
3. Mean X
4. Mean Y
5. Peak Z
6. Signal Magnitude Area (SMA)
7. Mean Z

Applying this subset of features, rather than the full set of features, on a HNM scheme yielded the following results:

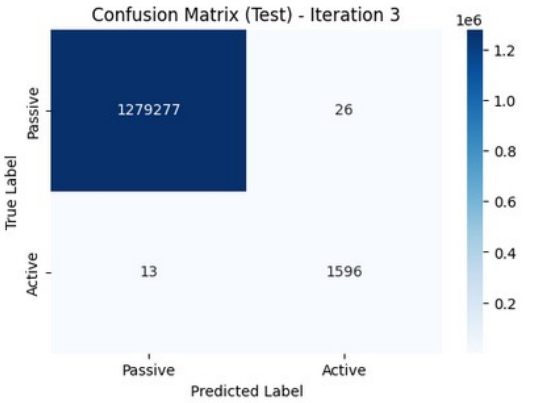


Figure I.8 Confusion matrix for fence intrusion detection top classifier using feature importance with HNM for four iterations

Metric	Value
Accuracy	1.0000
Precision	0.9840
Recall	0.9919
F1-Score	0.9879

Table I.2 Quantitative metrics for fence intrusion detection top classifier using feature importance with HNM for four iterations

Since the feature importance analysis reveals that a single feature accounts for more than 99% of the model's predictive power, we determined that further evaluation of feature selection methods, such as SFS or BO, is unnecessary. These methods would likely yield the same dominant feature, making additional computational effort redundant.
Using this subset of features found with feature importance, we can now optimize for decision tree hyperparameters and see if further improvements are possible.
A list of the hyperparameters optimized over can be found in 4.5.
By applying Bayesian Optimization over the hyperparameter space for 50 iterations, we find the following hyperparameters:

- **Criterion**: `gini`
- **Splitter**: `best`

- **Max Depth**: 18
- **Min Samples Split**: 2
- **Min Samples Leaf**: 1
- **Max Features**: `sqrt`

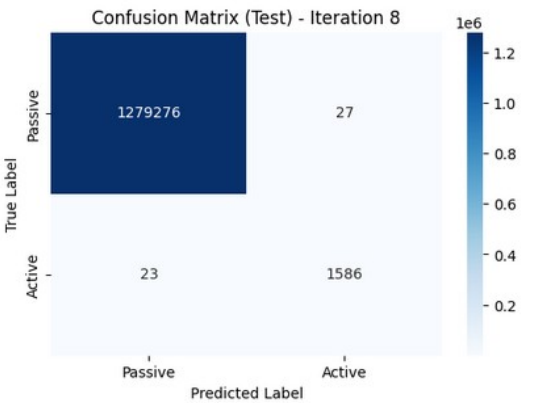


Figure I.9 Confusion matrix for fence intrusion detection top classifier using feature importance and optimized hyperparameters with HNM for eight iterations

Metric	Value
Accuracy	1.0000
Precision	0.9833
Recall	0.9857
F1-Score	0.9845

Table I.3 Quantitative metrics for fence intrusion detection top classifier using feature importance and optimized hyperparameters with HNM for eight iterations

Bottom classifier. For the base model that uses all features, HNM did not improve results. Therefore, the presented results for the bottom classifier using all features are without HNM.

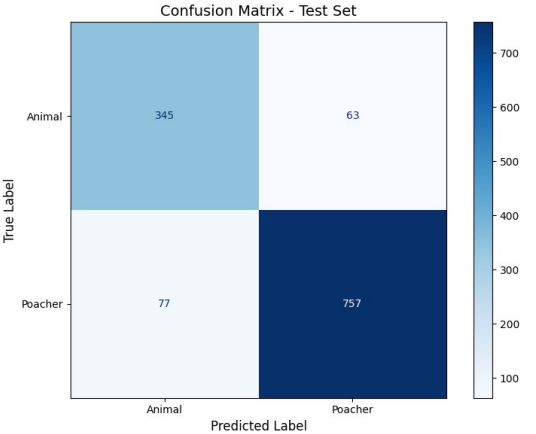


Figure I.10 Confusion matrix for fence intrusion detection bottom classifier using all features.

Metric	Value
Accuracy	0.8873
Precision	0.9232
Recall	0.9077
F1-Score	0.9154

Table I.4 Quantitative metrics for fence intrusion detection bottom classifier using all features.

Applying feature importance as feature selection method yielded the following results:

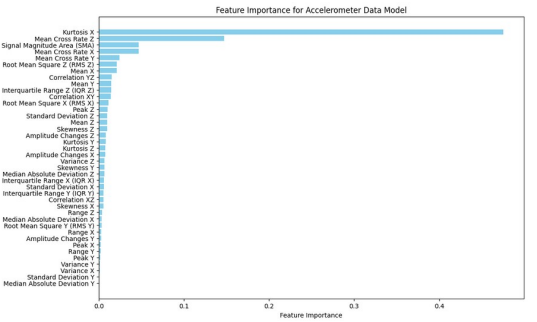


Figure I.11 Plot of feature importance for fence intrusion detection bottom classifier.

By cutting of features with less than 0.001 importance, in other words features that contribute to less that 0.1 % of the decision making capabilities, the features left become:

1. Kurtosis X
2. Mean Cross Rate Z
3. Signal Magnitude Area
4. Mean Cross Rate X
5. Mean Cross Rate Y
6. Root Mean Square Z
7. Mean X
8. Correlation YZ
9. Mean Y
10. Interquartile Range Z
11. Correlation XY
12. Root Mean Square X
13. Peak Z
14. Standard Deviation Z

Applying this subset of features, rather than the full set of features, on a HNM scheme yielded the following results:

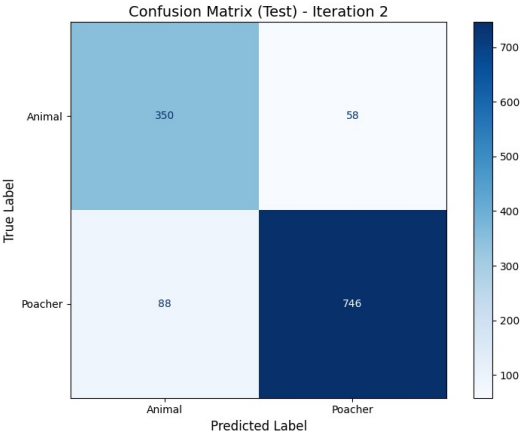


Figure I.12 Confusion matrix for fence intrusion detection bottom classifier using feature importance with HNM for two iterations.

Metric	Value
Accuracy	0.8824
Precision	0.9279
Recall	0.8945
F1-Score	0.9109

Table I.5 Quantitative metrics for fence intrusion detection bottom classifier using feature importance with HNM for two iterations.

Applying backwards SFS and optimizing for precision yielded the following results:

- Standard Deviation X
- Root Mean Sqauare Z (RMS Z)
- Range X
- Mean Cross Rate X
- Mean Cross Rate Z

Applying this subset of features, rather than the full set of features, on a HNM scheme yielded the following results:

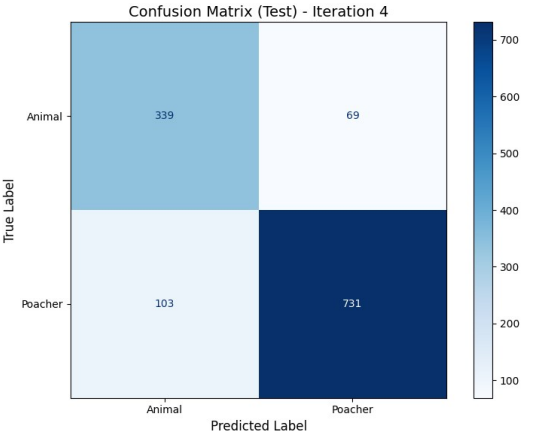


Figure I.13 Confusion matrix for fence intrusion detection bottom classifier using sequential feature selection with HNM for four iterations.

Metric	Value
Accuracy	0.8615
Precision	0.9137
Recall	0.8765
F1-Score	0.8947

Table I.6 Quantitative metrics for fence intrusion detection bottom classifier using sequential feature selection with HNM for four iterations.

Applying bayesian optimization as a feature selection method yielded the following results:
The following results and optimization curve were found using Bayesian optimization for feature selection, optimizing for FPR:

- Mean X
- Mean Y
- Mean Z
- Median Absolute Deviation Y
- Median Absolute Deviation Z
- Standard Deviation X
- Standard Deviation Y

- Variance Z
- Signal Magnitude Area
- Root Mean Square X
- Root Mean Square Y
- Root Mean Square Z
- Interquartile Range X
- Interquartile Range Y
- Range Y
- Range Z
- Peak X
- Peak Y
- Peak Z
- Mean Cross Rate X
- Mean Cross Rate Z
- Kurtosis Y
- Skewness X
- Skewness Y
- Amplitude Changes X

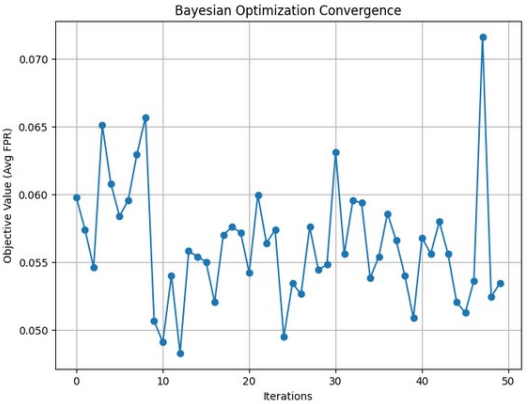


Figure I.14 Optimization curve of bayesian optimization, minimizing FPR for the bottom classifier by changing the feature selection for 50 iterations

Applying this subset of features, rather than the full set of features, on a HNM scheme yielded the following results:

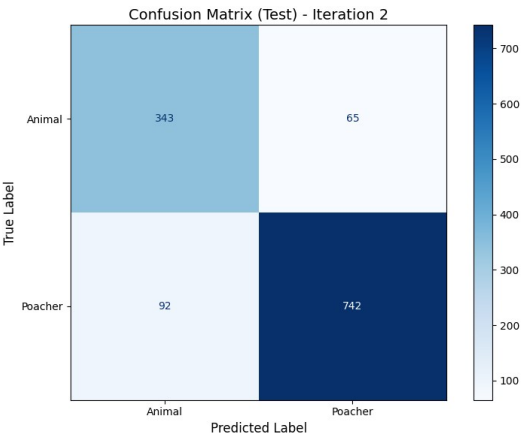


Figure I.15 Confusion matrix for fence intrusion detection bottom classifier using bayesian optimization with HNM for two iterations.

Metric	Value
Accuracy	0.8736
Precision	0.9195
Recall	0.8897
F1-Score	0.9043

Table I.7 Quantitative metrics for fence intrusion detection bottom classifier using bayesian optimization with HNM for two iterations.

I.3 ROC-AUC

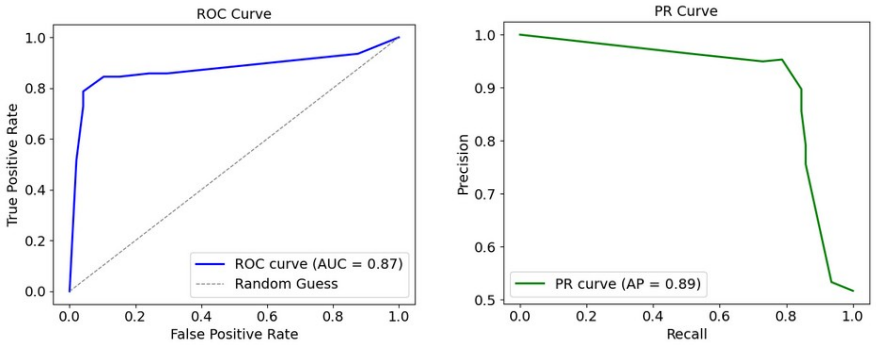
A brief note is made here on an additional metric that the authors recommend for future work.
The ROC-AUC (Receiver Operating Characteristic - Area Under Curve) metric is popular for binary classifiers. It will not be used in this thesis for the sake of brevity, however it is presented here as it illustrates the crux of our problem - attaining a good level of recall without sacrificing precision. It plots the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds and measures the area under the curve. The true positive rate (TPR) is calculated as

$$\text{TPR} = \frac{TP}{TP+FN},$$

and the false positive rate (FPR) is calculated as

$$\text{FPR} = \frac{FP}{FP+TN}.$$

A classification threshold is a measure of how certain we want to be in order to classify our data as the target class. When we increase the threshold, we make the model more conservative, meaning it predicts the class as the target only when it is more confident. This improves precision but makes recall worse. Conversely, when we decrease the threshold, we create a more liberal model with better recall but worse precision. This is called the Precision-Recall (PR) trade-off.
This highlights a general trade-off in machine learning models: as we try to increase TPR, we tend to increase FPR as well. With a perfect model, we would like to achieve a high TPR regardless of the FPR, that is to say, a large area under the TPR-FPR curve.



(a) ROC Curve **(b)** PR Curve

Figure I.16 Comparison of ROC and PR curves for a decision tree classifier. The ROC curve illustrates the trade-off between the true positive rate and false positive rate, while the PR curve highlights the balance between precision and recall.

J

Images taken of work at Makulu



Figure J.1 The two authors demounting the Doogee S100 from a tree.



Figure J.2 One of the authors collecting active gunshot data.



Figure J.3 One of the authors putting on an accelerometer at the Makulu fence.

K

Laptop Hardware Specifications

Attribute	Value
Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Address sizes	39 bits physical, 48 bits virtual
Byte Order	Little Endian
CPU(s)	8
On-line CPU(s) list	0-7
Vendor ID	GenuineIntel
Model name	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
CPU family	6
Model	140
Thread(s) per core	2
Core(s) per socket	4
Socket(s)	1
Stepping	1
CPU(s) scaling MHz	28%
CPU max MHz	4200.0000
CPU min MHz	400.0000
BogoMIPS	4838.40

Table K.1 System Specifications

L

Evidence of poacher breaching fence at Makulu



Figure L.1 Evidence of a poacher exiting the Makulu property by bending the fence open in the same way that was chosen in 6.4 and was made part of Case 3.



Figure L.2 A footprint from a poacher crouching and exiting the fence in the opening pictured above

Bibliography

[1] J. Kamminga, E. Ayele, N. Meratnia, and P. Havinga. "Poaching detection technologies—a survey". *Sensors* **18**:5 (2018). Published: 8 May 2018, p. 1474. DOI: 10.3390/s18051474. URL: https://www.mdpi.com/1424-8220/18/5/1474.

[2] Qulinda. *Qulinda official website*. [Online; accessed October 2024]. n.d. URL: https://www.qulinda.com.

[3] Smart Savannahs. *Smart savannahs official website*. [Online; accessed October 2024]. n.d. URL: https://www.smartsavannahs.org.

[4] D. Grahn and T. Cooper. *Gunshot Detection and Direction of Arrival Estimation Using Machine Learning and Received Signal Power*. LiTH-ISY-EX–23/5594–SE. Master's Thesis. Linköping University, Linköping, Sweden, 2023.

[5] D. Goderik and A. Westlund. *Detection and Tracking of Elephants using Seismic Direction of Arrival Estimates*. LiTH-ISY-EX–23/5590–SE. Master's Thesis. Linköping University, Linköping, Sweden, 2023.

[6] I. C. Duporge. *Advancing Remote Sensing Methods to Monitor Wildlife*. A thesis submitted for the degree of Doctor of Philosophy. Supervisors: Professor David W. Macdonald, Adjunct Professor Tiejun Wang. Assessors: Professor Martin Wikelski, Professor Adrian Thomas. PhD thesis. University of Oxford, Lady Margaret Hall, 2021.

[7] E. Castelli. *Enhancing Anti-Poaching Efforts Through Predictive Analysis of Animal Movements and Dynamic Environmental Factors*. Master of Science in Computing Science specialization Artificial Intelligence, 120 Credits. Supervisors: Frank Dignum, Keith Larson. Examiner: Juan Carlos Nieves Sanchez. MA thesis. Umeå Universitet, 2023.

[8] Google Developers. *Machine learning crash course*. [Online; accessed November 2024]. n.d. URL: https://developers.google.com/machine-learning/crash-course.

[9] Scikit-learn Developers. *Plot feature importances in random forest*. [Online; accessed November 2024]. n.d. URL: https://scikit-learn.org/1.5/auto_examples/ensemble/plot_forest_importances.html.

[10] GeeksforGeeks. *Sequential feature selection (sfs)*. [Online; accessed November 2024]. n.d. URL: https://www.geeksforgeeks.org/sequential-feature-selection/.

[11] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon. *Bayesian optimization is superior to random search for machine learning hyperparameter tuning: analysis of the black-box optimization challenge 2020*. 2021. arXiv: 2104.10201 [cs.LG]. URL: https://arxiv.org/abs/2104.10201.

[12] P. I. Frazier. "A tutorial on bayesian optimization". *arXiv preprint arXiv:1807.02811* (2018). [Online; accessed November 2024]. URL: https://doi.org/10.48550/arXiv.1807.02811.

[13] C. Hvarfner, E. O. Hellsten, and L. Nardi. "Vanilla bayesian optimization performs great in high dimensions". *arXiv preprint arXiv:2402.02229* (2024). [Online; accessed November 2024]. URL: https://doi.org/10.48550/arXiv.2402.02229.

[14] Lund University. *Minor field studies*. [Online; accessed November 2024]. n.d. URL: https://www.lunduniversity.lu.se/current-students/international-opportunities/grants-and-scholarships-studies-and-traineeships-abroad/minor-field-studies#about.

[15] Wikipedia contributors. *Savanna*. https://en.wikipedia.org/wiki/Savanna. Accessed: 2024-12-12. 2024.

[16] T. Kuiper. *The reliability and conservation value of ranger-collected data on elephant poaching*. A dissertation submitted for the degree of Doctor of Philosophy, Supervised by Professor E.J. Milner-Gulland. PhD thesis. University of Oxford, 2020.

[17] Wikipedia. *Semi-structured interview*. [Online; accessed November 2024]. n.d. URL: https://en.wikipedia.org/wiki/Semi-structured_interview.

[18] United Nations Office on Drugs and Crime (UNODC). *Wildlife crime worth usd 8-10 billion annually, ranking it alongside human trafficking, arms and drug dealing in terms of profits: unodc chief*. [Online; accessed November 2024]. 2014. URL: https://www.unodc.org/unodc/en/frontpage/2014/May/wildlife-crime-worth-8-10-billion-annually.html.

[19] Wildlife Justice Commission. *A Preliminary Analysis of Raw Rhino Horn Prices in Africa and Asia*. [Accessed November 2024]. 2019. URL: https://wildlifejustice.org/wp-content/uploads/2019/02/FINAL-raw-rhino-horn-digital-1.pdf.

[20] Wildlife Justice Commission. *Rapid Assessment of the Illegal Ivory Trade in 2020*. [Accessed November 2024]. 2020. URL: https://wildlifejustice.org/wp-content/uploads/2020/08/WildlifeJusticeCommission_Rapid-Assessment-Of-The-Illegal-Ivory-Trade-in-2020_August2020_Spreads.pdf.

[21] Convention on International Trade in Endangered Species of Wild Fauna and Flora (CITES). *Species Specific Matters: Rhinoceroses (Rhinocerotidae spp.)* CoP19 Doc. 75. [Accessed November 2024]. Nineteenth meeting of the Conference of the Parties, Panama City, Panama, 2022. URL: https://cites.org/sites/default/files/documents/E-CoP19-75.pdf.

[22] S. B. Nesar, B. M. Whitaker, and R. C. Maher. "Machine learning analysis on gunshot recognition". In: *2024 Intermountain Engineering, Technology and Computing (IETC)*. [Accessed November 2024]. IEEE, Bozeman, Montana, USA, 2024. URL: https://doi.org/10.1109/IETC61393.2024.10564263.

[23] S. A. Qureshi, L. Hussain, H. M. Alshahrani, S. R. Abbas, M. K. Nour, and N. Fatima. "Gunshots localization and classification model based on wind noise sensitivity analysis using extreme learning machine". *IEEE Access* **10** (2024). [Accessed November 2024]. URL: https://doi.org/10.1109/ACCESS.2022.3198966.

[24] M. S. Kabir, J. Mir, C. Rascon, M. L. U. R. Shahid, and F. Shaukat. "Machine learning inspired efficient acoustic gunshot detection and localization system". *University of Wah Journal of Computer Science* **3**:1 (2022). [Accessed November 2024], p. 40. URL: https://uwjcs.org.pk/index.php/ojs/article/view/40.

[25] M. Iskander. "Chapter 3 - geotechnical underground sensing and monitoring". In: S. Pamukcu et al. (Eds.). *Underground Sensing*. Academic Press, 2018, pp. 141–202. ISBN: 978-0-12-803139-1. DOI: https://doi.org/10.1016/B978-0-12-803139-1.00003-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128031391000035.

[26] P. Sjövik and E. Wahledow. *Detection and Localization of Elephants using a Geophone Network*. MA thesis. Linköping University, 2022.

[27] C. O'Connell-Rodwell, B. Arnason, and L. Hart. "Seismic properties of asian elephant (elephas maximus) vocalizations and locomotion". *The Journal of the Acoustical Society of America* **108** (2001), pp. 3066–72. DOI: 10.1121/1.1323460.

[28] J. R. Hutchinson, D. Schwerda, D. J. Famini, R. H. I. Dale, M. S. Fischer, and R. Kram. "The locomotor kinematics of asian and african elephants: changes with speed and size". *Journal of Experimental Biology* **209**:19 (2006), pp. 3812–3827. ISSN: 0022-0949. DOI: 10.1242/jeb.02443.

eprint: https://journals.biologists.com/jeb/article-pdf/209/19/3812/1541905/3812.pdf. URL: https://doi.org/10.1242/jeb.02443.

[29] J. Wood, C. O'Connell-Rodwell, and S. Klemperer. "Methodological insights: using seismic sensors to detect elephants and other large mammals: a potential census technique". *Journal of Applied Ecology* **42** (2005), pp. 587–594. DOI: 10.1111/j.1365-2664.2005.01044.x.

[30] A. Szenicer, M. Reinwald, B. Moseley, T. Nissen-Meyer, Z. Mutinda Muteti, S. Oduor, A. McDermott-Roberts, A. G. Baydin, and B. Mortimer. "Seismic savanna: machine learning for classifying wildlife and behaviours using ground-based vibration field recordings". *Remote Sensing in Ecology and Conservation* **8**:2 (2022), pp. 236–250. DOI: https://doi.org/10.1002/rse2.242. eprint: https://zslpublications.onlinelibrary.wiley.com/doi/pdf/10.1002/rse2.242. URL: https://zslpublications.onlinelibrary.wiley.com/doi/abs/10.1002/rse2.242.

[31] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. *Squeezenet: alexnet-level accuracy with 50x fewer parameters and <0.5mb model size*. 2016. arXiv: 1602.07360 [cs.CV]. URL: https://arxiv.org/abs/1602.07360.

[32] *SM-24 Geophone Element*. I/O Sensor Nederland bv. 2006.

[33] D. Cieslak and N. Chawla. "Learning decision trees for unbalanced data". In: 2008, pp. 241–256. ISBN: 978-3-540-87478-2. DOI: 10.1007/978-3-540-87479-9_34.

[34] L. Raileanu and K. Stoffel. "Theoretical comparison between the gini index and information gain" (2001).

[35] S. Bates, T. Hastie, and R. Tibshirani. "Cross-validation: what does it estimate and how well does it do it?" *Journal of the American Statistical Association* **119**:546 (2023), pp. 1434–1445. ISSN: 1537-274X. DOI: 10.1080/01621459.2023.2197686. URL: http://dx.doi.org/10.1080/01621459.2023.2197686.

[36] M. W. Hayward and G. I. Kerley. "Fencing for conservation: restriction of evolutionary potential or a riposte to threatening processes?" *Biological Conservation* **142**:1 (2009). [Accessed November 2024], pp. 1–13. URL: https://doi.org/10.1016/j.biocon.2008.09.022.

[37] C. E. Wilkinson, A. McInturff, M. Kelly, and J. S. Brashares. "Quantifying wildlife responses to conservation fencing in East Africa". *Biological Conservation* **256** (2021). [Accessed November 2024], p. 109071. URL: https://doi.org/10.1016/j.biocon.2021.109071.

[38] P. A. Lindsey, C. L. Masterson, A. L. Beck, and S. S. Romañach. "Ecological, social and financial issues related to fencing as a conservation tool in Africa". In: *Fencing for Conservation*. [Accessed November 2024]. Springer, New York, 2012, pp. 215–234. DOI: 10.1007/978-1-4614-0902-1_12. URL: https://doi.org/10.1007/978-1-4614-0902-1_12.

[39] G. Kriel. *Fence-sitter warning system, how it works*. Accessed: 2024-12-09. 2015. URL: https://www.farmersweekly.co.za/agri-technology/farming-for-tomorrow/fence-sitter-warning-system/.

[40] A. Yousefi, A. A. Dibazar, and T. W. Berger. "Intelligent fence intrusion detection system: detection of intentional fence breaching and recognition of fence climbing". In: *2008 IEEE Conference on Technologies for Homeland Security*. [Accessed November 2024]. IEEE, 2008. DOI: 10.1109/THS.2008.4534415. URL: https://doi.org/10.1109/THS.2008.4635057.

[41] J. Ahokas. *Automated animal behavior analysis using accelerometer activity tags*. LiTH-ISY-EX-24/5688-SE. Master's Thesis. Linköping University, Department of Electrical Engineering, Division of Automatic Control, Linköping, Sweden, 2024.

[42] W. Hämäläinen, M. Järvinen, P. Martiskainen, and J. Mononen. "Jerk-based feature extraction for robust activity recognition from acceleration data". *2011 11th International Conference on Intelligent Systems Design and Applications* (2011). [Accessed November 2024]. DOI: 10.1109/ISDA.2011.6121760. URL: https://doi.org/10.1109/ISDA.2011.6121760.

[43] SparkFun Electronics. *Sparkfun openlog artemis*. [Online; accessed November 2024]. 2024. URL: https://www.sparkfun.com/products/16832.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
February 2025

Document Number
TFRT-6269

Author(s)
Jakob Olof Popov Wirén
Kasper Nordenram

Supervisor
Kristian Soltesz, Dept. of Automatic Control, Lund University, Sweden
Bo Bernhardsson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Machine Learning for Anti-Poaching: Decision Tree Applications on the Savannah

Abstract

The use of technology, specifically machine learning, to address complex issues in developing regions has become increasingly prominent. In the context of wildlife conservation, combating poaching has emerged as a significant application of such technology.
This research project aims to develop machine-learning-based low-cost antipoaching solutions. It presents three case studies, each exploring a unique sensor modality and use-case on the savannah, while maintaining a common framework of machine learning algorithms.
1. The first use case is the detection of gunshots using microphone-based systems.
2. The second use case is the detection of elephant footsteps using geophonebased systems.
3. The third use case is the detection of perimeter fence intrusions using accelerometer-based systems.
Each sensor modality is accompanied by a decision-tree algorithm that aims to optimize the Probability of Detection (POD)-to-False Positive Rate (FPR) ratio. Decision trees are computationally low-cost and intuitive machine learning models that are suitable for systems with low power and memory. The data used in this thesis was collected by the authors at a wildlife sanctuary in South Africa.
The project established high-performing baselines that achieved favorable POD to FPR ratios across two out of three cases. Additionally, the project provided the complete dataset and all machine learning results, including the selected features, the most effective feature selection methods, and an extensive analysis informed by local knowledge from interviewed rangers. This comprehensive approach aids the future professional development of a practically useful system tailored for the savannah environment.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-187

Recipient's notes

Security classification

http://www.control.lth.se/publications/