University of Essex

School of Computer Science & Electronic Engineering

Meteor Detection & Classification

MSc Dissertation Report

Author: **Ashwin Purushothama Dhas**
email: **ap23710@essex.ac.uk**
Registration Number: **2320993**

Supervisor: **Dr. Adrian F Clark**

December 11, 2024

Abstract

This project introduces MIDAS (Meteor Identification and Detection Automation System), a machine learning system for finding meteor objects in spectrogram images. Four methods were tested: YOLO11m, MeteorNet, a Hybrid model (using MobileNet_v3 and Random Forest), and a Fusion method that combined the results of YOLO11m and MeteorNet. The system was trained using data from the Belgian RAdio Meteor Stations (BRAMS). Among all methods, the Fusion model worked the best, giving better accuracy and coverage by using YOLO11m's ability to detect clearly and MeteorNet's strength in finding faint meteors. McNemar's test confirmed that the Fusion model was the most reliable method. A web application was also made to show the detections, making it easier for users to use. In the future, the plan is to add more training data, make the system faster, and test MIDAS in other meteor stations to increase its use in atmospheric research.

Contents

List of Figures

List of Tables

Preface

This dissertation reflects the author's interest in meteors and the use of machine learning in atmospheric science. With a background in Aeronautics, the author naturally connected with this project, which combines concepts from aeronautics and artificial intelligence. The project was inspired by the work of the Belgian RAdio Meteor Stations (BRAMS), whose freely available spectrogram data formed the basis of this research.

The research faced challenges, such as labelling individual objects in spectrogram images and building a custom machine learning model. These difficulties offered valuable learning experiences, helping the author gain a deeper understanding of meteor detection and the practical use of machine learning methods.

Special thanks go to Dr Adrian F. Clark for his guidance and support during the research. The author is also grateful to the BRAMS team for their contributions to the field and for keeping the spectrogram data accessible to everyone. Heartfelt appreciation is extended to the author's family for their constant support and encouragement throughout this journey.

The author hopes this dissertation will advance the field of meteor detection and inspire further research on applying machine learning to atmospheric science.

CHAPTER 1

Introduction

Meteor detection is very important in astronomy research. It helps us understand how meteoroids behave, what they are made of, and how they move when they enter Earth's atmosphere. Meteoroids are small pieces of space debris that burn up in the atmosphere, creating bright streaks of light called meteors. When this happens, they ionize the air around them and leave trails that reflect radio waves. These reflections, called meteor echoes, allow scientists to learn more about the size, shape, and paths of meteoroids.

The Belgian RAdio Meteor Stations (BRAMS), managed by the Belgian Institute for Space Aeronomy (BIRA-IASB), are an example of advanced meteor detection systems. BRAMS uses a method called forward-scattering radio, where radio waves are sent out and bounce off meteor trails, creating signals with Doppler shifts. This method allows meteors to be tracked at any time of the day and in any weather. However, the large amount of spectrogram data produced is difficult to analyze manually, which is why automated systems are needed.

New developments in machine learning, especially deep learning, have changed how data is analyzed in science. Neural Networks, which are known for being very accurate in recognizing images and patterns, are a good option for automating meteor detection. These models can quickly process large amounts of data, find patterns in spectrogram images, and detect meteor echoes even when there is noise or interference.

This project, called MIDAS (Meteor Identification and Detection Automation System), focuses on building and testing machine learning models to automatically detect meteors using BRAMS spectrogram data. The work started by downloading spectrogram images from the BRAMS database and manually labeling meteor echoes to create a reliable training dataset. Four methods were tested: YOLO11m, a high-performing object detection model; MeteorNet, a custom neural network designed for this project; a Hybrid model (MobileNet_v3 with Random Forest); and a Fusion model that combined YOLO11m and MeteorNet for better results. A web app was also made to let users upload spectrogram images and get automated meteor detection results.

This dissertation is organized as follows. Chapter 2 looks at past research on meteor detection and how machine learning has been used in this area. Chapter 3 discusses legal, ethical, and social aspects of the project. Chapter 4 explains how MIDAS was built, including how data was prepared, models were developed, and everything was connected. Chapter 5 is a user guide for MIDAS, explaining how to install it, use it, and understand its results. Chapter 6 shows how well the models worked, with pictures and statistical analysis to confirm the findings. Finally, Chapter 7 summarizes the main results and suggests ideas for improving meteor detection and using machine learning in the future.

By improving automated meteor detection, this project helps us understand more about meteoroids and how they interact with Earth's atmosphere. It provides useful tools for scientists and supports progress in both atmospheric science and machine learning.

CHAPTER 2

Literature Review

Meteor detection has greatly improved over time, moving from manual observation methods to advanced techniques using machine learning. This chapter looks at past research, showing the progress from traditional methods to modern uses of machine learning and object detection in studying meteors.

2.1 Optical Meteor Detection

One of the first methods used to detect meteors was optical techniques, which depended on taking photographs and making visual observations. Hawkes and Jones (1975)[2] used long-exposure photography to study meteor showers, giving detailed information about how often meteors occur, their brightness, and how they are spread out in space. While these studies provided useful information about meteor showers, they had some limitations:

- They required clear weather and darkness to work.
- They could not capture faint meteors or those hidden by clouds.
- They needed a lot of manual work to collect and analyze data.

Even though these methods helped lay the foundation for meteor research, their limitations in accuracy and scalability led to the development of new techniques, such as using radio waves and automated systems.

2.2 Radio Meteor Detection

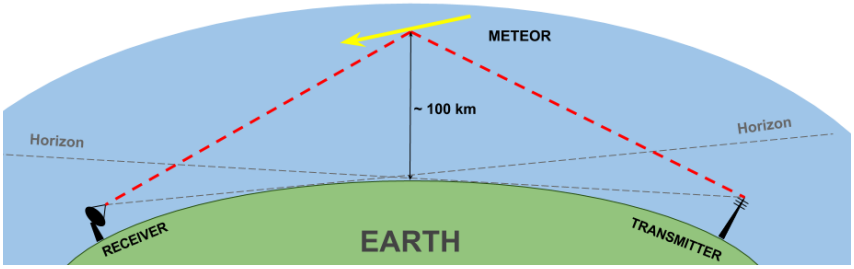

Figure 2.1: Illustration of forward scattering radio observation used in meteor detection [1].

The introduction of radio-based methods brought a major change in meteor detection. McKinley (1961)[3] introduced forward-scattering techniques, where radio waves from the ground bounce off ionized meteor trails, creating Doppler-shifted signals. This method made it possible to observe meteors continuously, no matter the weather or time of day. Wislez and Verbeeck (2006)[4] showed how well forward scattering works using the BRAMS network. These advancements allowed large-scale monitoring with very little manual effort and made it possible to detect meteors that optical methods couldn't see because they were too small or not bright enough.

$$f_D = \frac{2 \cdot v_r \cdot f_0}{c}$$

Where:
f_D: Doppler shift.
v_r: Radial velocity of the meteor relative to the observer.
f_0: Frequency of the transmitted radio wave.
c: Speed of light in a vacuum.

Wislez and Verbeeck (2006)[4] showed how well forward scattering works by using the BRAMS network. Their research explained how ionized meteor trails act as temporary reflectors, making it possible to detect meteor activity accurately. Radio-based systems also provided the following benefits:

- They allowed large-scale monitoring with very little manual effort.
- They could detect meteors that optical methods couldn't see because of their small size or low brightness.

However, radio meteor detection produces a huge amount of data, making it necessary to use advanced tools like spectrogram analysis and machine learning to understand and analyze the information effectively.

2.3 Spectrogram Analysis

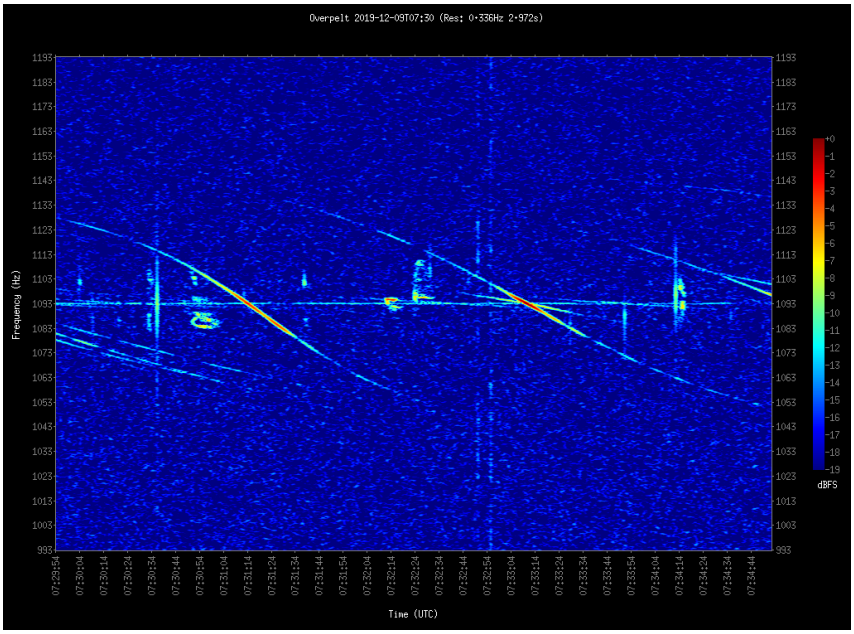

Figure 2.2: spectrogram obtained from BRAMS at BEOUDS on March 26 2011 at 00:40 UT

Spectrograms provide a frequency-time representation of radio signals, enabling detailed analysis of meteor echoes. The Fast Fourier Transform (FFT) converts raw radio data into spectrograms, revealing Doppler shifts caused by meteors [5]. The Fourier Transform is expressed as:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft}\, dt$$

Where:
$X(f)$: Frequency domain representation.
$x(t)$: Time domain signal.
f: Frequency.
t: Time.

Lunsford (2009)[5] highlighted the usefulness of FFT (Fast Fourier Transform) for detecting faint meteor echoes even in noisy backgrounds. The BIRA-IASB developed custom software for processing BRAMS data, which improved spectrogram analysis and made it easier to detect meteor trails accurately. However, manually interpreting spectrograms is still slow and can lead to mistakes. Research like Lampert et al. (2009)[20] looked into automated line detection in spectrogram images, opening the door for machine learning methods to make the analysis process faster and more efficient.

2.4 Machine Learning in Meteor Detection

2.4.1 Emergence of Machine Learning

Machine learning (ML) has changed object detection by automating the analysis of complicated data. ML models can find patterns in noisy data, making them a good choice for detecting meteors in spectrograms. Earlier ML methods focused on extracting features and classifying them, but with advancements in deep learning [6], systems can now learn directly from raw data. These newer methods reduce the need for manual feature selection and improve detection accuracy.

2.4.2 Traditional Models in Image Object Detection

Random Forest, introduced by Breiman (2001)[6], is a flexible machine learning algorithm that uses multiple decision trees to improve prediction accuracy and reduce overfitting. Breiman's research highlighted how Random Forest can be understood easily, as it shows which features are most important, making it a popular choice for scientific studies. Random Forest has been used in many fields, such as environmental monitoring and astronomy, where both accuracy and clarity are important. For example, Zhang et al. (2019)[7] used Random Forest to classify celestial objects and achieved strong results even with noisy data. This ability to handle noise makes Random Forest a good choice for detecting meteors in spectrogram images. Its classification process is based on:

$$P(c) = \frac{1}{T} \sum_{t=1}^{T} p_t(c)$$

Where:
$P(c)$: Final class probability.
T: Number of trees in the forest.
$p_t(c)$: Class probability predicted by tree t.

Support Vector Machines (SVM), introduced by Cortes and Vapnik (1995)[8], is another traditional machine learning method widely used for object detection and classification tasks.

SVM works by finding the best hyperplane that separates data points from different classes in a high-dimensional space. The mathematical formula for the decision function in SVM is:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b\right)$$

Where:

- x_i: Training data points.
- y_i: Class labels.
- α_i: Lagrange multipliers.
- $K(x_i, x)$: Kernel function.
- b: Bias term.

SVM has been commonly used in object detection because it works well with high-dimensional data and performs effectively on small datasets. For example, Zhai et al. (2021)[9] developed an SVM-based meteor detection system that used spectrogram features for classification, achieving high accuracy and reliability. The use of different kernel functions, like linear, polynomial, and radial basis function (RBF), makes SVM flexible enough to handle different types of data and patterns. Recent improvements have made SVM even more useful. Combining SVM with techniques like Histogram of Oriented Gradients (HOG) or embeddings from deep learning has been shown to give better results on complex datasets. This is especially helpful for meteor detection, where the data might have overlapping or noisy signals. SVM is easy to understand and works well on new, unseen data, making it a valuable tool in traditional object detection methods. It works well with ensemble methods like Random Forest by providing another way of separating and classifying data.

2.4.3 Neural Networks

Calders et al. (2019)[10] used Convolutional Neural Networks (CNNs) to detect meteor echoes in spectrograms. Their model used sliding windows to find areas with high confidence and then improved these areas with post-processing methods like blurring and thresholding. Even though noise made training difficult, the model achieved a precision of 0.852, recall of 0.676, and an F1-score of 0.754. Peña-Asensio et al. (2023)[19] used CNNs with Gradient-weighted Class Activation Mapping (Grad-CAM) for meteor detection. Their method achieved an impressive 98% precision and was able to separate meteors from static objects like clouds and buildings. These studies show how CNNs are useful for working with complex spectrogram data.

2.4.4 YOLO Framework

The "You Only Look Once" (YOLO) framework, introduced by Redmon et al. (2016)[17], changed object detection by treating it as a single regression problem, making real-time processing possible. YOLO splits images into grids and predicts bounding boxes and class probabilities

at the same time, making object detection both fast and accurate. The bounding box predictions are defined as:

$$b_x = \sigma(t_x) + c_x,$$
$$b_y = \sigma(t_y) + c_y,$$
$$b_w = p_w e^{t_w},$$
$$b_h = p_h e^{t_h}$$

Where:
σ: Sigmoid function.
c_x, c_y: Top-left coordinates of the grid cell.
p_w, p_h: Dimensions of the prior bounding box.

The Intersection over Union (IoU) metric is used to evaluate bounding box accuracy:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Later versions of YOLO have improved its performance and accuracy. For example, Zhuang et al. (2017)[18] used YOLOv8 for object detection and achieved high precision and recall. YOLO's ability to process large datasets quickly in real-time makes it a great option for detecting meteor echoes in spectrogram images. The latest version, YOLO11, brings new design improvements that make it better and more reliable for object detection tasks. These updates show how the YOLO framework keeps getting better and more useful for solving different detection problems, including meteor detection.

2.4.5 U-Net Framework

The U-Net model, first introduced by Ronneberger et al. (2015)[11], was originally made for biomedical image segmentation but is now used in many other fields, including meteor detection. Its encoder-decoder design, with skip connections, helps in accurate localisation and segmentation tasks. Unlike regular object detection models, U-Net works on pixel-level classification, making it good for identifying object trails in spectrogram images. Ronneberger et al.[11] showed that U-Net can work well with small datasets by using data augmentation techniques. Its symmetrical structure keeps high-resolution spatial details from the encoder and uses them in the decoder. This makes U-Net especially useful for finding and segmenting object trails in complex spectrograms, where noise can hide the target.

In object detection, changes to the U-Net design have been tried to improve its performance. For example, Lampert et al. (2009)[20] tested line detection methods in spectrogram images. These findings led to modifications in U-Net, like changing the kernel sizes or adding attention mechanisms, to better separate meteor trails from background noise. U-Net is also flexible enough to include specific modules, like dilated convolutions, which increase the receptive field without losing resolution. This flexibility shows U-Net's potential for meteor segmentation, solving problems like low signal-to-noise ratios and the varying appearances of meteors.

2.4.6 MobileNet Framework

MobileNet, introduced by Howard et al. (2017)[12], is a group of lightweight convolutional neural networks designed to work efficiently on mobile and small devices. It uses depthwise

separable convolutions, which reduce the amount of computation needed while still giving good results in feature extraction tasks. This makes MobileNet a good choice for applications with limited resources. The latest version, MobileNet_v3, includes new features like squeeze-and-excitation modules and a special activation function called Hard-Swish. Howard et al. (2019)[12] showed that these updates improve accuracy while keeping the model lightweight. This mix of efficiency and good performance makes MobileNet_v3 great for extracting features from images, especially when saving computational power is important.

2.4.7 Hybrid Approaches

Hybrid models that combine object detection and segmentation are becoming more popular. Cascade R-CNN (Cai & Vasconcelos, 2018)[14] improves detection results by refining them through multiple stages. Its updated version, Cascade Mask R-CNN (2023), adds attention modules to make segmentation more accurate. Panoptic Segmentation models, like Panoptic FPN (Kirillov et al., 2019[15]) and its newer versions (2023), combine instance and semantic segmentation into one system. These methods are useful for meteor detection, where it is important to locate objects and segment them accurately at the same time.

2.4.8 Application of Pre-trained Models

Pre-trained models are now very important for quickly creating and using machine learning solutions. EfficientDet, introduced by Tan & Le (2020), now has an updated version called EfficientDet-DX (2023), which includes vision transformers for better understanding of context while staying efficient. New models like SAM (Segment Anything Model, 2024) have also been developed, offering zero-shot segmentation. These models allow users to segment object trails without needing large labeled datasets, saving a lot of time and effort during model training.

2.5 Labelling Techniques for Object Detection

Accurate and efficient labelling of datasets is very important for object detection tasks. Well-annotated data helps machine learning models learn useful patterns and perform accurately during predictions. Many techniques and tools have been created to make the labelling process easier, especially for object detection in images.

2.5.1 Manual Labelling Tools

Manual labelling is still the best way to create high-quality datasets because human annotators can accurately identify and mark objects. Tools like LabelMe, introduced by Russell et al. (2008)[22], are commonly used for labelling images in computer vision tasks. LabelMe provides an easy-to-use interface where annotators can draw boxes or polygons around objects, making it flexible and accurate for preparing datasets.

2.5.2 Semi-Automatic and Automatic Labelling

For larger datasets, semi-automatic and automatic labelling methods can help save a lot of time and effort. These methods use pre-trained models or algorithms to create initial labels, which are then improved by human annotators.

- Interactive Labelling Tools: Tools like VIA (VGG Image Annotator) (Dutta et al., 2016[21]) provide semi-automatic labelling with easy editing options for objects that are already

pre-labelled.

- Model-Assisted Labelling: Pre-trained models, such as YOLO11m or U-Net, can create rough bounding boxes or segmentations. Human annotators then adjust these predictions to make them more accurate, saving time while keeping good quality.
- Crowdsourcing Platforms: Platforms like Amazon Mechanical Turk and Labelbox allow many people to work together on labelling tasks. While this is fast, proper checks are needed to make sure the labels are consistent and accurate.

2.6 Summary

This chapter discussed the progress of meteor detection, starting from optical and radio-based methods to advanced machine learning techniques. Using machine learning in meteor detection systems has greatly improved accuracy and efficiency. This project builds on these improvements by using YOLO11m, a custom U-Net model, and a hybrid method to automate and enhance meteor detection in spectrogram images.

CHAPTER 3

Legal, Ethical, and Related Issues

This chapter looks at the legal, ethical, and other issues related to developing and possibly using MIDAS (Meteor Identification and Detection Automation System). It talks about how the project affects health and safety, follows legal rules, meets societal and ethical responsibilities, sticks to industry standards, and supports sustainability. It also considers the possibility of MIDAS being used by research organisations like BRAMS (Belgian RAdio Meteor Stations) under BIRA-IASB (Belgian Institute for Space Aeronomy).

3.1 Health and Safety Considerations

Most of the work for MIDAS was done on the author's personal laptop and local computing setups. The author followed health and safety rules to avoid problems from long hours in front of the screen, like taking regular breaks and setting up an ergonomic workspace. For training and testing the machine learning models, high-performance computing (HPC) systems were used remotely, which reduced risks linked to working on-site. The author made sure to follow all safety guidelines from the institution while using the HPC systems. This project does not involve working directly with meteor detection hardware, but it supports such systems indirectly. MIDAS aims to automate and simplify tasks, reducing the manual work researchers need to do. This helps lower health risks from long hours of manual annotation and data processing.

3.2 Legal Considerations

The development of MIDAS followed all academic and professional rules. The spectrogram data used in the project came from BRAMS, a public network for meteor research. The author made sure this data was used only for academic purposes and followed the terms of use set by BRAMS and BIRA-IASB. The author used open-source tools and libraries like TensorFlow, PyTorch, YOLO11m, and LabelMe to create MIDAS. These tools were used according to their licenses and rules. Publicly available tools for labelling and machine learning were also included in the project, ensuring everything followed the proper usage policies. The name MIDAS was chosen by the author to match the software's purpose and follow academic naming rules. Since MIDAS was developed as part of an academic project, it has not been made into a commercial product. This ensures it works well with existing research systems in BRAMS and other academic institutions, without causing any legal or regulatory issues.

3.3 Cultural, Societal, and Related Considerations

The development of MIDAS does not involve working with people or using sensitive personal data, so there are no ethical issues related to privacy or data protection. However, the author thought carefully about the larger societal and ethical impact of using automated detection systems in scientific research. Transparency was identified as an important ethical aspect. The author documented all methods, workflows, and assumptions used in developing MIDAS to ensure that others can reproduce the work and hold it accountable. Transparency in

machine learning is especially important in fields like astronomy, where research results add to shared scientific knowledge.

If institutions like BRAMS use MIDAS, there are extra ethical responsibilities for the software's design. Apart from meeting technical needs, MIDAS was designed to follow the ethical standards of the research community. To address concerns about interpretability, the author included features to visualize and check model predictions, allowing researchers to verify results and spot any biases. Accessibility was also a key focus. MIDAS was made to work on commonly available hardware, so researchers with limited resources can still use it. This supports fairness in research, making advanced tools available to more people and encouraging inclusive scientific progress.

3.4 Ethical Considerations

The author followed ethical principles throughout the project, ensuring transparency, honesty, and integrity at every stage of development. Ethical concerns in the MIDAS project mainly focused on the responsible use of machine learning. The author took steps to prevent the misuse of the software for anything other than meteor detection by keeping detailed documentation and clearly explaining its purpose. Reproducibility was an important part of developing MIDAS. The author recorded all methods and hyperparameters so that other researchers could repeat the work and get the same results. This commitment to openness follows good practices in academic research and highlights the ethical duty to support the scientific community.

3.5 Adherence to Standards

While developing MIDAS, the author followed industry standards and best practices to make sure it is reliable, easy to maintain, and compatible with other systems. The software follows the Python PEP 8 coding standard, which helps keep the code clear and easy for other developers to understand and work with. All machine learning models were trained and tested using common metrics like Precision, Recall, IoU (Intersection over Union), and F1 Score, following standard practices in object detection.

The data annotation process used the LabelMe tool, which creates annotations in a format that works with YOLO11m models. Using this widely accepted annotation format ensures that the data can be used in other projects or shared with collaborators.

If MIDAS is adopted by institutions, its modular design makes it easy to integrate with existing systems. The software works with Python libraries like TensorFlow and PyTorch, keeping it compatible with modern research tools. The deployment part, built with Streamlit, follows modern web application standards, making it simple and user-friendly.

3.6 Sustainability

The author focused on sustainability while developing MIDAS. The software is designed to run on regular computers, so it doesn't need expensive GPUs or servers. This helps reduce both the cost and environmental impact of using it. MIDAS automates tasks like manual annotation and data processing, which usually take a lot of time and resources. By making these processes faster and easier, the software helps improve the efficiency and sustainability of meteor research.

If BRAMS or other institutions use MIDAS, it can support long-term data analysis and

monitoring without needing much extra infrastructure. The use of open-source libraries and tools also makes it easier for the community to improve the software, keeping it sustainable over time. Using automated systems like MIDAS supports global sustainability goals, such as improving education and encouraging innovation. By making advanced detection tools available to more people, the project helps make scientific research more accessible and promotes sustainable growth in knowledge.

CHAPTER 4

Design and Implementation of MIDAS

The Meteor Identification and Detection Automation System (MIDAS) was created by the author to solve the challenges of identifying meteors in spectrogram images from the Belgian RAdio Meteor Stations (BRAMS). These spectrograms include meteor echoes, interference, and reflections, making classification difficult. To handle these challenges, MIDAS uses advanced machine learning techniques like YOLO11m, MeteorNet, and a Random Forest model to detect meteor objects accurately. This chapter explains how MIDAS was designed and implemented, including how the dataset was prepared, how the software works, and how the annotation process was done using LabelMe.

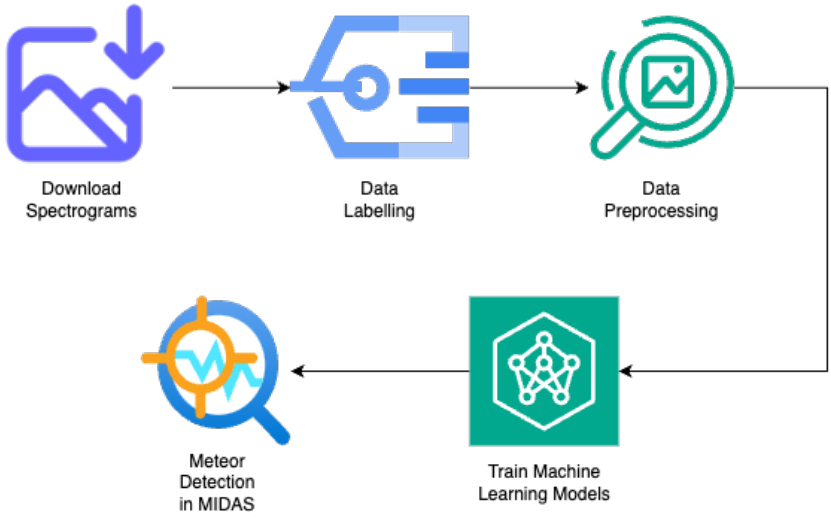

Figure 4.1: Objectives to build MIDAS

4.1 System Objectives

The author set several main goals for developing MIDAS to make sure it meets the needs of researchers and improves meteor detection. These goals are:

- Platform Independence: MIDAS is designed to work smoothly on Windows, Linux, and macOS. This cross-platform support ensures that researchers and collaborators can use it without needing to make platform-specific changes.
- Ease of Use: MIDAS includes a simple Graphical User Interface (GUI) made with Streamlit. This allows researchers to use MIDAS easily, even with little technical knowledge. Users can upload spectrogram images and choose detection models through an easy-to-use interface.
- Model Options and Testing: MIDAS uses different machine learning models, including YOLO11m for object detection, MeteorNet for segmentation, and a Random Forest classifier with MobileNet for feature extraction and patch-based classification. These options ensure strong detection performance.

- Annotation-Based Training: Accurate annotations are important for model performance. MIDAS uses pre-annotated spectrogram datasets for training, with meteor echoes manually labeled using tools like LabelMe.
- Scalability and Automation: MIDAS can process large datasets efficiently and supports data augmentation to expand training data. This helps the models generalize better and become more reliable.

4.2 Data Collection, Filtering, and Preparation

4.2.1 BRAMS Spectrogram Characteristics

Spectrogram images posed several unique challenges that needed careful attention during the design and implementation of MIDAS. These challenges came from the complexities of the BRAMS spectrograms, which include meteor echoes, noise, and interference, each with its own features.

- Dynamic Power Range: BRAMS spectrograms use a color-coded scale to show power levels in decibels relative to full scale (dBFs). The range of these power levels changes due to noise levels and bright pixels. This variability required preprocessing to make the input data consistent. Although MIDAS does not directly scale the dBFs values, the changes in power levels were considered during data preparation and training.
- Meteor Echoes: Meteor echoes, the main focus of MIDAS, appear as distinct features in spectrogram images. They are divided into two types:
  - Underdense Meteor Echoes: These are short vertical lines that last for a fraction of a second. They are hard to confirm clearly because the time resolution of spectrograms is about 3 seconds. To analyze these echoes, the frequencies they cover must be separated from other frequencies in the raw data to get a clean meteor profile.
  - Overdense Meteor Echoes: These last longer and have more complex shapes. In one sample spectrogram, five overdense meteor echoes were found. Their longer presence makes them easier to identify, but their irregular shapes add difficulty to the annotation and detection process.
- Noise and Interference: Spectrograms also contain noise and interference, which make meteor detection harder. Main types of interference include:
  - Beacon Signal at 1093 Hz: A horizontal line at 1093 Hz represents the beacon frequency, transmitted either directly or through the atmosphere. Its stability depends on the receiver's local oscillator (LO). For example, with an ICOM-R75 receiver, the LO stability changes with temperature, causing slight frequency shifts. With RSP2 receivers locked to a GPDO reference, this signal stays fixed at 1000 Hz.
  - Broad-band Interference: Vertical interrupted lines, such as those around 07:32:50 UT in the sample spectrogram, show broad-band interference. These signals cover the full 200 Hz range, extending from 900 Hz to over 1.7 kHz, making them easy to separate from meteor echoes.

– Airplane Reflections: Long-lasting "S-shaped" echoes come from airplane reflections. These shapes are caused by the double Doppler effect due to the airplane's movement relative to the beacon and the receiver. Straight-moving planes create classic S-shapes, while planes changing directions make more complex patterns. In one sample spectrogram, eight airplane reflections were recorded.

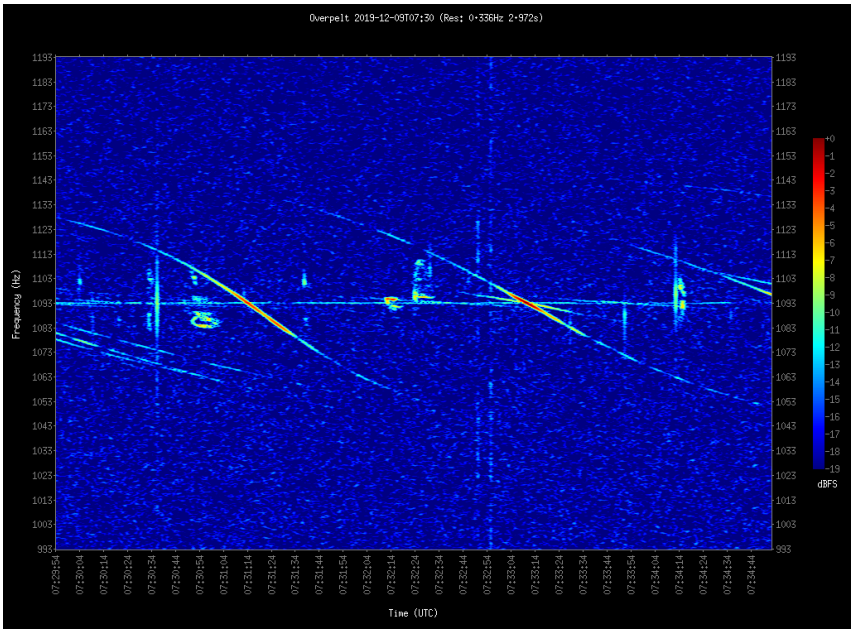

Figure 4.2: spectrogram obtained from BRAMS at OVERPELT on December 09 2019 at 07:30 UT

To make the detection process simpler, only meteor echoes were annotated in the dataset. This was done to reduce computational effort and prevent confusing predictions caused by overlapping bounding boxes. These factors influenced the data annotation and model training steps, allowing MIDAS to handle the complexities of BRAMS spectrogram images effectively.

4.2.2 Data Collection

The author downloaded the spectrogram images for the MIDAS project using the BRAMS Data Downloader Tool[23] dating from 09/09/2024 to 10/09/2024. The downloader interface allowed the author to select specific locations for data collection. These selected locations provided a wide range of meteor observation data, ensuring the availability of diverse and high-quality spectrograms for the project.

A total of approximately 8000 spectrogram images were downloaded from the chosen locations. The wide selection of observation points ensured comprehensive data coverage, capturing various meteor echo characteristics across different geographical regions. The inclusion of multiple locations increased the likelihood of obtaining spectrograms with diverse meteor patterns and environmental noise profiles.

4.2.3 Data Filtering and Selection

From the collected 8000 spectrogram images, the author carefully filtered and identified 1300 images for labelling, training, and augmentation. The filtering process involved:

- Relevance Check: Images were reviewed to ensure they contained clear meteor echoes, avoiding spectrograms dominated by background noise or irrelevant signals such as airplane reflections or beacon signals.
- Diversity: A mix of images with underdense and overdense meteor echoes was selected to provide a balanced dataset that could represent varied meteor types.
- Clarity: Spectrograms with distinct and identifiable meteor echoes were prioritized to improve the effectiveness of labelling and model training.

4.2.4 Data Preparation

The 1300 selected spectrogram images formed the base dataset for labelling and augmentation. These images were annotated using LabelMe, an open-source annotation tool, to create precise bounding boxes around meteor echoes. After labelling, the dataset was augmented using horizontal and vertical flipping techniques, increasing the number of training samples to 3400. This augmentation process was essential for enhancing the generalization capabilities of the models and ensuring they performed well on varied meteor patterns.

By carefully selecting and preparing the dataset, the author ensured that the models trained for MIDAS were robust and capable of handling the complexities of meteor detection in spectrogram images.

4.2.5 Data Annotation with LabelMe

LabelMe, an open-source annotation tool, was used a lot to create ground-truth labels. The process included:

- Annotation Workflow: The author manually identified meteor echoes in spectrogram images. Bounding boxes were drawn around the meteor echoes, and their coordinates were saved in .txt files compatible with YOLO11m. Only meteor echoes were annotated because of their scientific importance. Overlapping bounding boxes for other objects, like airplane reflections, were avoided to keep the predictions clear.
- Challenges: Annotating spectrogram images for meteor detection was challenging, especially when deciding whether to include all objects or focus only on meteor echoes. Including all objects, like airplane reflections, broad-band interference, and beacon signals, created cluttered annotations with overlapping bounding boxes, making the spectrograms visually confusing. This clutter made it hard for the author to focus on meteor features. On the other hand, annotating only meteor echoes provided a simpler approach, reducing visual clutter and making the annotations clearer. This focus helped the authot concentrate on meteors, the main objects of interest, without being distracted by other irrelevant features. Annotating only meteor echoes also made the model predictions easier to understand by avoiding overlapping detections for unrelated objects. Special attention was given to distinguishing between underdense and overdense meteor echoes, which differ in duration and complexity. This method ensured a cleaner dataset, improved model accuracy, and gave researchers a more useful and effective tool for studying meteor spectrograms. The annotation format includes the class, the bounding box center coordinates (x, y), and the width and height of the bounding box, all normalized to the image dimensions.

- File Management: Annotated files were organized using flipping augmentation techniques (horizontal and vertical) and stored in different folders for training, validation, and testing. Augmentations were applied to the original dataset, increasing the samples from 1300 to 3400. This helped the model learn better and handle different meteor shapes and orientations.

- An example annotation in a label file is as follows:

  0 0.62890625 0.48862019914651494 0.0068359375 0.07112375533428165
  Where:
  0: the class ID, which specifies "meteor".
  0.62890625: the normalized x coordinate of the center of the bounding box, relative to the width of the image.
  0.48862019914651494: the normalized y coordinate of the center of the bounding box, relative to the height of the image.
  0.0068359375: the normalized width of the bounding box, relative to the width of the image.
  0.07112375533428165: the normalized height of the bounding box, relative to the height of the image.

- A sample of an image with all object annotations and an image with only meteor annotations are as follows.

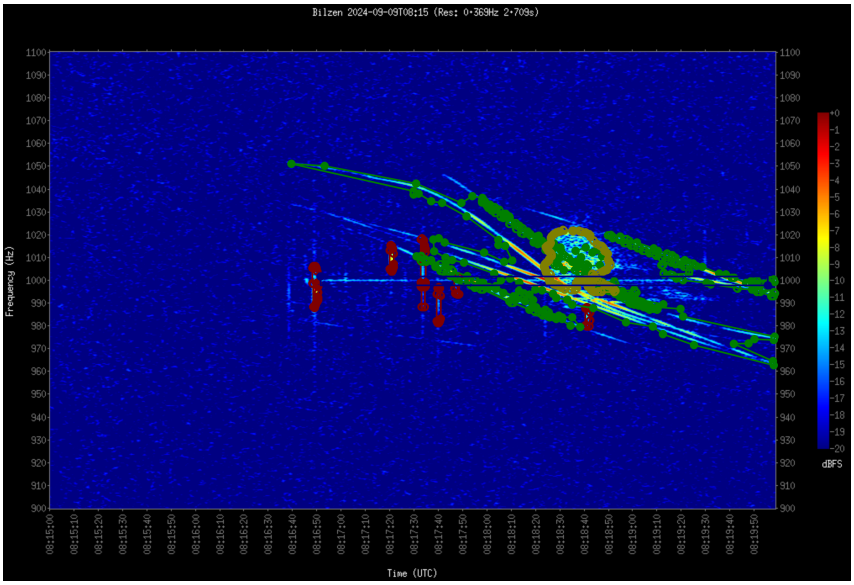

Figure 4.3: spectrogram image with all object annotations

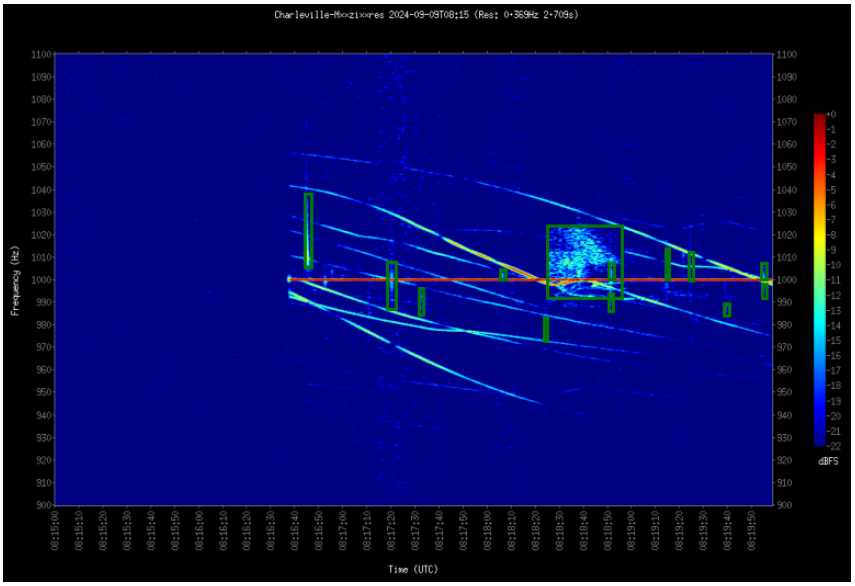

Figure 4.4: spectrogram image with meteor annotation

4.2.6 Data Splitting: Training, Validation, and Testing

The dataset for MIDAS was split into 3,000 samples for training, 200 for validation, and 200 for testing to ensure a proper approach to model development and evaluation. The training dataset, being the largest, was used to help the models learn patterns related to meteor echoes, including augmented samples for better learning. The validation dataset was used to adjust hyperparameters and check for overfitting using early stopping. The testing dataset, kept completely separate, was used to check the final performance of the models, ensuring fair evaluation and accurate benchmarking. This split made sure the models were trained and validated properly while maintaining high evaluation standards.

4.3 Model Development and Integration

4.3.1 YOLO11m

As part of the MIDAS (Meteor Identification and Detection Automation System) project, the author utilized the YOLO11m object detection model to identify meteor objects in spectrogram images. YOLO11m, a state-of-the-art object detection algorithm, was selected due to its efficiency and high performance in detecting objects within complex visual data.

Model Configuration

The YOLO11m model was loaded using the Ultralytics Python package, which provided a robust framework for model training, evaluation, and deployment. The following configuration was used during the training process:

- Model Architecture: YOLO11m variant
- Pretrained Weights: The model utilized pre-trained weights (YOLO11m.pt) as a starting point.
- Optimization Algorithm: Stochastic Gradient Descent (SGD)

- Learning Rate: Initial learning rate set to 0.001 with cosine annealing for adjustment.
- Batch Size: 16
- Image Size: 960x960 pixels
- Augmentation Techniques:
- Auto augment: autoaugment
- Flip horizontally: 50
- Flip vertically: 50
- Scale: 0.2
- Loss Functions: Configured for single-class detection with a confidence threshold of 0.3 and Intersection over Union (IoU) threshold of 0.6.
- Input Image Size: 960x960 pixels.
- Learning Rate: 0.001 with a cosine learning rate scheduler.
- Augmentations: Auto augmentations, horizontal flips (fliplr), and vertical flips (flipud).
- Training Epochs: 300 epochs with early stopping after 10 epochs of no improvement.

Model Training

The dataset for training the YOLO11m model included spectrogram images labeled with meteor and background annotations. The original dataset had 1300 samples, which was increased to 3400 samples using horizontal and vertical flipping for augmentation.

The model was trained three times with different numbers of training samples (3000, 2000, and 1000). While the main goal was to train the model with 3000 samples, the author tested how YOLO11m's performance changes when the training dataset size is different. After training, the model was validated and tested using the assigned subsets. Validation metrics were calculated during training, and testing was done after training was completed.

4.3.2 MeteorNet

The author created and implemented a custom neural network called MeteorNet to detect meteor objects in spectrogram images. MeteorNet uses a convolutional neural network (CNN) design made for binary segmentation tasks to identify meteor areas against the background. It was built using TensorFlow and Keras frameworks, focusing on efficiency, accuracy, and suitability for the dataset.

Model Architecture

MeteorNet's architecture was designed as a U-Net-like structure, optimized for binary segmentation:

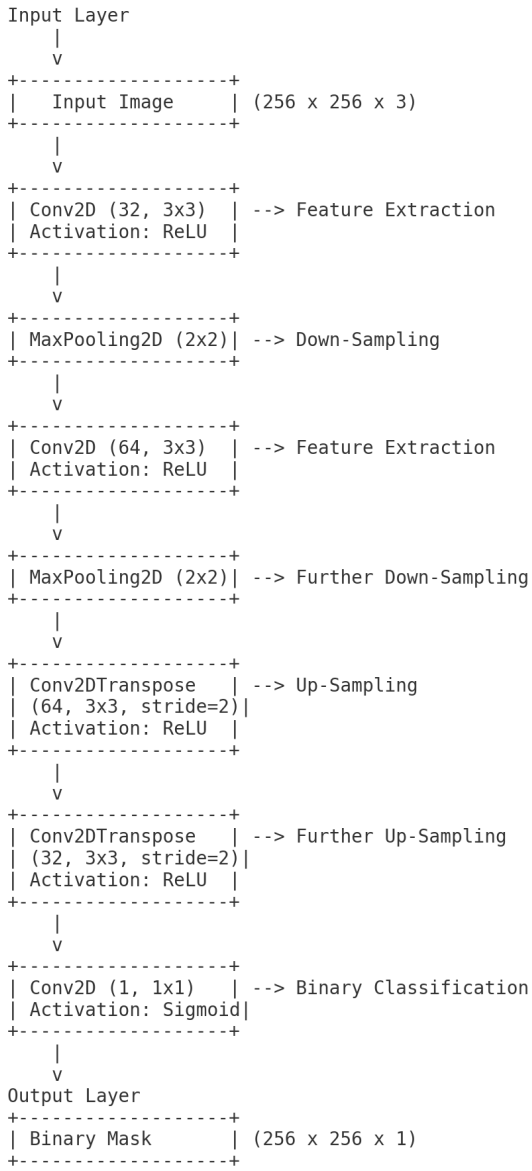

Figure 4.5: Architecture of MeteorNet

- Input Layer: Accepts spectrogram images resized to 256x256 pixels.
- Convolutional Layers: Extract features using filters of increasing depth (32, 64).
- Deconvolutional Layers: Reconstruct the meteor regions from extracted features.
- Output Layer: Generates a binary mask of meteor regions.

The model was compiled with the following metrics:

- Precision: To measure the exactness of detections.
- Recall: To evaluate the completeness of detections.
- F1-Score: A harmonic mean of precision and recall.

Data Preprocessing

- Image Processing:
  - Images were resized to the input shape defined in the configuration.
  - Normalized to a range of [0, 1] for compatibility with the neural network.
- Mask Creation:
  - Label files in YOLO11m format (bounding box coordinates) were converted into binary masks, where meteor regions were set to 1 and background regions to 0.
  - Masks were resized to match the dimensions of the input images and reshaped to include a single channel.

Model Training

The training process for MeteorNet was executed with the following configuration:

- Input Shape:
- $256 \times 256 \times 3$
- Loss Function: Binary cross-entropy
- Optimizer: Adam optimizer
- Metrics: Precision, Recall, Intersection over Union (IoU), Mean IoU, Area Under Curve (AUC), Custom F1-Score (computed at a specified threshold)
- Epochs: 50
- Batch Size: 16
- Early Stopping: Incorporated with a patience of 5 epochs to prevent overfitting.

The dataset used YOLO11m-format labels, which were converted into pixel-wise segmentation masks. The author wrote a script to read YOLO11m labels and create binary masks automatically. Although the main goal was to train MeteorNet on the full dataset (3000 samples), the author also trained the model on smaller subsets of 2000 and 1000 samples to check how the training dataset size affects segmentation performance.

4.3.3 Hybrid(Random Forest with MobileNet)

The author created a hybrid model that combined MobileNet_V3 as a feature extractor and a Random Forest classifier to detect meteor objects in spectrogram images. This method used MobileNet_V3's lightweight and strong feature extraction abilities along with Random Forest's easy-to-understand and reliable classification.

Model Architecture

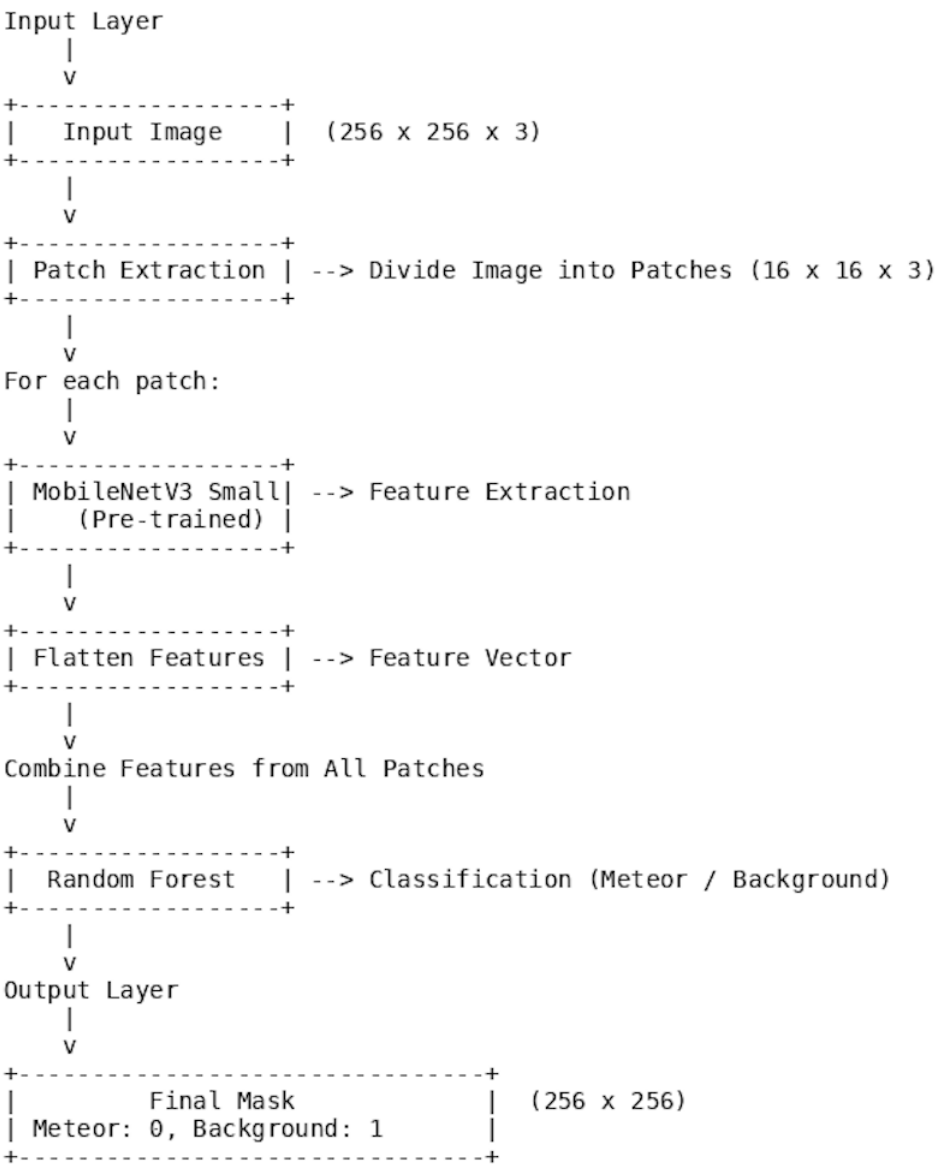

Figure 4.6: Architecture of hybrid model

The hybrid model consisted of:

- Feature Extractor: MobileNet_V3 (small variant) was used for feature extraction. The classifier layer of the pre-trained model was removed, leaving the convolutional layers to extract meaningful feature representations from spectrogram patches.
- Classifier: A Random Forest classifier was trained using the extracted features to classify patches into meteor or background categories.
- Configuration:
  - Number of Estimators: 50

  - Class Weight: Balanced
  - Random State: 42 for reproducibility

- Evaluation: Precision, Recall, F1-Score

Data Preprocessing

- Image Resizing: All spectrogram images were resized to 256×256 pixels for consistency with MobileNet_V3's input requirements.
- Patch Segmentation: Each resized image was divided into smaller patches of 16×16 pixels to localize processing.
- Feature Extraction: Each patch was passed through MobileNet_V3 to extract a flattened feature vector.
- Label Assignment: Patches overlapping with meteor bounding boxes were labeled as meteor (class 0), while non-overlapping patches were labeled as background (class 1).

Model Training

- Feature Extraction: Patches from spectrogram images were processed using MobileNet_V3, generating feature vectors.
- Random Forest Training: The extracted feature vectors and their corresponding labels were used to train the Random Forest classifier.

Because the Random Forest model had low accuracy and object detection performance, the author trained it only on the full dataset of 3000 samples. The hybrid model, which combined MobileNet_V3 for feature extraction and Random Forest for classification, was an experimental method in the MIDAS framework. Even though its accuracy and object detection performance were lower than YOLO11m and MeteorNet, this model showed that lightweight, pre-trained architectures can be used for feature extraction. The insights from this experiment helped in understanding the trade-offs of using patch-based classification for meteor detection.

4.4 Fusion Detection Technique

The Fusion Detection Technique combines the object detection capabilities of YOLO11m and the segmentation capabilities of MeteorNet to improve meteor detection accuracy in spectrogram images. The goal of this approach is to capitalize on YOLO11m's ability to locate meteor objects efficiently and MeteorNet's precision in segmenting meteor regions.

Configuration

The Fusion Detection Technique utilizes two models in tandem:

- YOLO11m Configuration:
  - Trained on meteor datasets for object detection.
  - Generates bounding boxes with confidence scores for meteor regions.
- MeteorNet Configuration:

- Trained on meteor datasets for segmentation-driven bounding box generation.
- Produces bounding boxes that localize meteor regions in the input image.

The outputs from both models are processed by a bounding box combiner, which merges overlapping or adjacent boxes into unified detections. This ensures accurate and consolidated localization of meteor objects within the spectrogram image.

Architecture

The Fusion Detection Technique processes spectrogram images through a pipeline integrating both YOLO11m and MeteorNet.

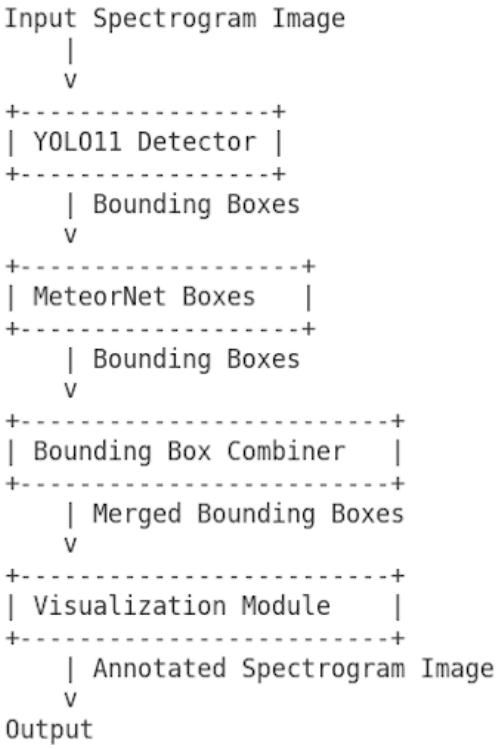

Figure 4.7: Architecture of fusion detection technique

First, YOLO11m detects possible meteor areas by creating bounding boxes with confidence scores. At the same time, MeteorNet processes the same image and produces bounding boxes based on its segmentation results, focusing on specific meteor areas. These bounding boxes are then merged using a custom bounding box combiner, which joins overlapping or nearby boxes to create a single detection. Finally, the combined bounding boxes are shown on the original spectrogram image, giving a clear view of the detected meteor objects.

4.5 GUI Implementation

To make it easy to interact with the MIDAS system and display the results of the models, the author created a Graphical User Interface (GUI) using Streamlit. The goal was to build a simple and efficient platform for showing meteor detection results, integrating four models:

YOLO11m, MeteorNet, Random Forest with MobileNet_V3-based feature extraction (Hybrid), and the new Fusion Detection Technique. The GUI was designed to be modular and scalable, making it suitable for demonstrating the features of MIDAS.

4.5.1 Application Design

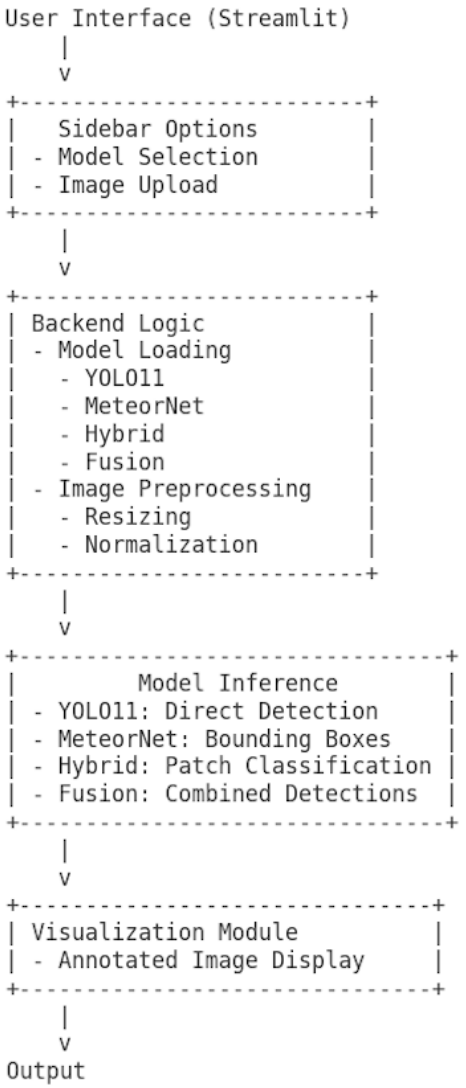

Figure 4.8: Design of GUI application

Planning

The author outlined the functionality and user flow, deciding on essential features such as model selection, image upload, and result visualization.

Modular Architecture

The GUI was encapsulated within a Python class (MeteorDetectionApp) that grouped all methods and logic, including model loading, inference, and result visualization. Each model (YOLO11m, MeteorNet, Hybrid, and Fusion) was implemented as a separate method to maintain a clean and modular codebase. This approach ensured scalability and ease of integration for additional models or features in the future.

Dynamic Model Integration

The GUI dynamically loaded the selected model only when needed, minimizing resource usage and allowing for flexibility in extending the system with additional models in the future. A unified interface was designed for handling all models, standardizing steps such as image preprocessing and result annotation.

- YOLO11m: Integrated using the Ultralytics library for direct detection.
- MeteorNet: Integrated with TensorFlow/Keras to handle segmentation-based detection.
- Random Forest: Integrated with PyTorch (for MobileNet_V3 feature extraction) and scikit-learn (for patch classification).
- Fusion: Integrated as a combination of YOLO11m and MeteorNet outputs using a bounding box merging algorithm.

Image Preprocessing

The GUI included preprocessing logic tailored for spectrogram images. Tasks such as resizing, normalization, and patch segmentation (for the Hybrid model) were implemented to ensure compatibility with all integrated models.

Unified Results Display

The GUI annotated meteor detections (bounding boxes or masks) and displayed the processed image directly, ensuring a consistent and user-friendly visualization experience across all models.

Testing

Each model was validated within the GUI to ensure accurate and seamless operation.

CHAPTER 5

Using MIDAS

MIDAS (Meteor Identification and Detection Automation System) is an interactive platform for detecting meteors in spectrogram images using four models: YOLO11m, MeteorNet, Hybrid, and Fusion. Built with Streamlit, MIDAS provides a simple and flexible system where researchers can upload spectrogram images, choose detection models, and view results with bounding box annotations. Currently, MIDAS works only on local machines. Users need to clone the repository, install the required dependencies, and run the application locally. This chapter explains how to set up MIDAS, describes the user interface and its features, and discusses the strengths and limitations of the system.

5.1 System Requirements

The implementation of MIDAS requires a local machine with the following specifications:

- Operating System: Windows, Linux, or MacOS
- Python Version: 3.8 or higher
- Hardware Requirements:
  - A machine with at least 4GB RAM
  - A GPU is recommended but not mandatory, as it significantly enhances the performance of certain models such as YOLO11m and MeteorNet.

5.2 Installation

To use MIDAS, users need to clone the repository and install the required dependencies. The setup involves the following steps:

1. Clone the MIDAS repository. **git clone <repository-link>**
2. Navigate to the project directory. **cd <repository-name>**
3. Install the dependencies using the provided requirements.txt file. **pip install -r requirements.txt**
4. Launch the application using Streamlit. **streamlit run midas.py**

5.3 Operating MIDAS

The GUI of MIDAS is simple and flexible. It has a sidebar for user inputs and a main display area for processing and showing results. In the sidebar, users can choose one of the four detection models (YOLO11m, MeteorNet, Hybrid, or Fusion) and upload spectrogram images. After the detection process is done, the main display area shows the results with annotated bounding boxes.

To operate MIDAS, users must follow these steps:

1. Model Selection: The user selects a detection model from the dropdown menu in the sidebar.
2. Image Upload: The user uploads a spectrogram image by either dragging and dropping the file or browsing through the system's file explorer.
3. Run Detection: The user clicks the Detect button to process the uploaded image with the selected model.
4. View Results: The detection results are displayed in the main area of the GUI. Detected meteors are highlighted with bounding boxes, and each detection model uses a distinct color for its annotations.

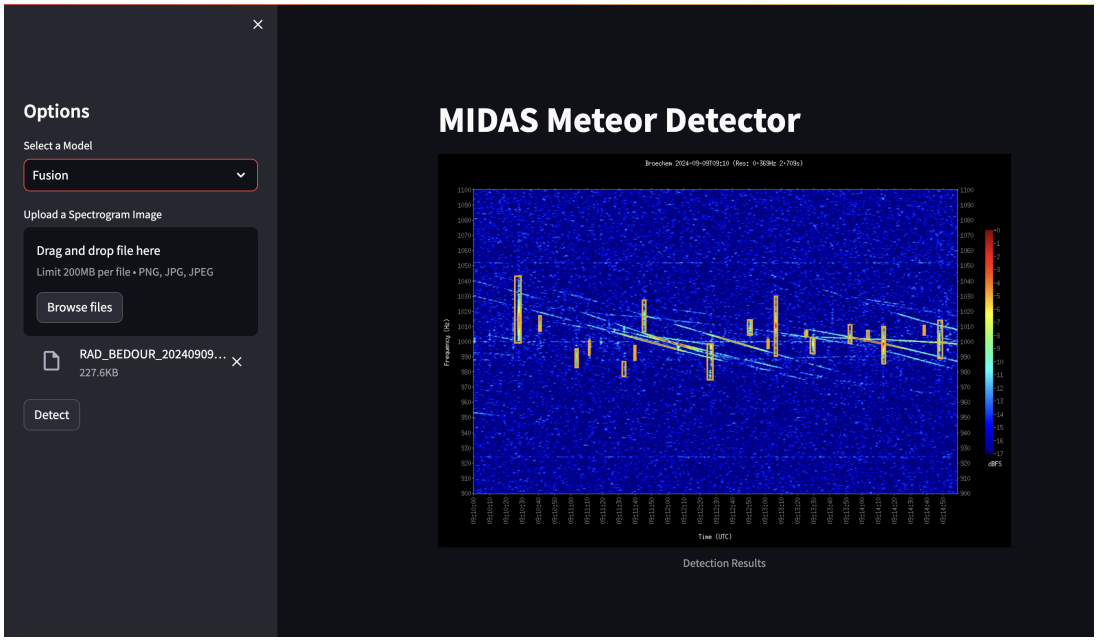

Figure 5.1: Screenshot of MIDAS GUI

5.4 Model Outputs

The four detection models integrated into MIDAS generate distinct outputs:

- YOLO11m: Outputs bounding boxes based on direct object detection, annotated in green.
- MeteorNet: Generates bounding boxes derived from segmentation, annotated in red.
- Hybrid Model: Combines MobileNet_V3 feature extraction with Random Forest classification to produce bounding boxes.
- Fusion Detection: Merges the outputs of YOLO11m and MeteorNet, integrating their bounding boxes into a unified set annotated in orange.

The results are visualized in the main display area of the GUI. Bounding boxes are overlaid on the spectrogram image to indicate detected meteors. Each model's outputs are color-coded, enabling users to differentiate between the detection results of each technique. The Fusion model merges overlapping detections to minimize redundancies and enhance accuracy.

5.5 Limitations

MIDAS currently operates only on local machines and requires manual installation of dependencies. Although a GPU is not mandatory, the absence of GPU support may lead to slower performance for computation-intensive models such as YOLO11m and MeteorNet. The author also advices a machine with atleast 4GB of RAM.

5.6 Future Scope

The author plans to improve MIDAS in future versions to fix current issues and make it more useful. One possible upgrade is to host MIDAS on a cloud platform or web server so that users can access it remotely. This would remove the need for local setup and dependency installation, making it easier for more people to use. The author also sees the possibility of connecting MIDAS with existing meteor observation networks like BRAMS (Belgian RAdio Meteor Stations). Adding MIDAS to BRAMS could make it a powerful tool for automated meteor detection and classification, improving the network's observation abilities and helping with faster and better data analysis in meteor science.

CHAPTER 6

Results and Evaluation

The results and evaluation chapter gives a detailed analysis of how well the developed models performed in detecting meteors. The evaluation uses key metrics like precision, recall, F1 score, and mean Average Precision (mAP). Each model is evaluated separately and then compared using McNemar's test.

6.1 YOLO11m Model Performance

The YOLO11m model, designed for detecting objects in spectrogram images, was tested with different dataset sizes to check its performance and reliability. The model was trained three times using datasets of 3000, 2000, and 1000 samples, with a fixed validation size of 200 samples and a test size of 200 samples. This section focuses on the results of YOLO11m trained on three datasets.

6.1.1 Performance of YOLO11m Trained on 3000 Samples

The YOLO11m model trained on 3000 samples showed strong performance on most metrics. It achieved a precision of 0.79351, recall of 0.75771, mAP50 of 0.81145, and mAP50-95 of 0.43244. The F1-Score reached 0.78, showing a good balance between precision and recall. These results highlight that the model generalized well and had strong detection capabilities when trained on a larger dataset.

6.1.2 Performance of YOLO11m Trained on 2000 Samples

When trained on 2000 samples, the YOLO11m model performed slightly less than the 3000-sample model. It achieved a precision of 0.79924, recall of 0.7568, mAP50 of 0.80037, and mAP50-95 of 0.43021. The F1-Score peaked at 0.77, showing it maintained a good balance between precision and recall. While the model still performed well, the smaller dataset reduced its reliability compared to the 3000-sample model.

6.1.3 Performance of YOLO11m Trained on 1000 Samples

The YOLO11m model trained on 1000 samples gave the best overall performance in several metrics. It achieved a precision of 0.79053, recall of 0.76746 (the highest among all configurations), mAP50 of 0.80458, and mAP50-95 of 0.4346 (the highest among all configurations). The F1-Score reached 0.78, tying with the 3000-sample model. These results show that the model performed well and generalized effectively even with a smaller dataset, making it the most efficient configuration overall.

6.1.4 Comparative Analysis

The performance of YOLO11m across the three configurations is visually summarized in the graphs and table below:

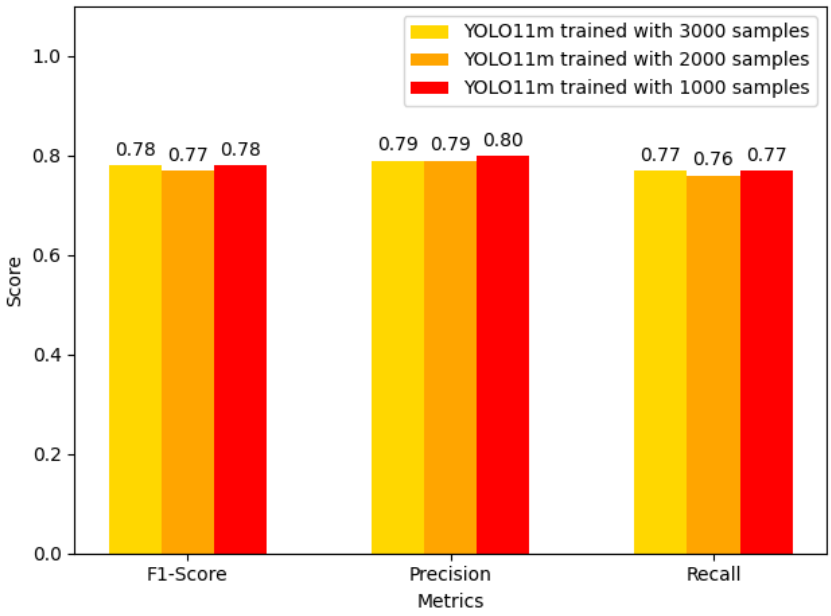

Figure 6.1: Performance Metrics for YOLO11m Trained on Different Sample Sizes

Metric	3000 Samples	2000 Samples	1000 Samples
Precision	0.79351	0.79924	0.79053
Recall	0.75771	0.7568	0.76746
mAP50	0.81145	0.80037	0.80458
mAP50-95	0.43244	0.43021	0.4346
F1-Score	0.78	0.77	0.78

Table 6.1: Performance Metrics for YOLO11m Trained on Different Sample Sizes

Based on the evaluation, the YOLO11m model trained on 1000 samples gave the most balanced performance. It performed best in recall and mAP50-95 while achieving a high F1-Score, showing its effectiveness even with a smaller dataset. The model trained on 3000 samples had a similar F1-Score and the highest mAP50 but was slightly lower in recall and mAP50-95, showing a trade-off in performance. The model trained on 2000 samples had the lowest overall performance. It gave stable results but was less reliable and consistent compared to the other two configurations.

6.2 MeteorNet Model Performance

The MeteorNet model, designed for segmentation-based meteor detection, was tested with different dataset sizes to study its performance and adaptability. The model was trained with 3000, 2000, and 1000 samples, while the validation set was kept at 200 samples and the test set at 200 samples. This section looks at how MeteorNet performed with these dataset sizes to find the best setup for this project.

6.2.1 Performance of MeteorNet Trained on 3000 Samples

The MeteorNet model trained on 3000 samples showed the most balanced performance. Its precision was 0.5885, which was lower than the models trained on smaller datasets, but the

recall improved significantly to 0.4988, showing better coverage of positive instances. The IoU and mean IoU stayed at 0.9976 and 0.4988, respectively. The AUC was the highest among all setups, at 0.9925, showing excellent reliability. The F1-Score was also the highest, at 0.5408, showing a good balance between precision and recall.

6.2.2 Performance of MeteorNet Trained on 2000 Samples

When trained with 2000 samples, MeteorNet had a slight drop in precision to 0.6089, but the recall improved slightly to 0.4820, showing better coverage of positive instances. The IoU and mean IoU were the same as with 3000 samples, at 0.9976 and 0.4988, respectively. The AUC went up slightly to 0.9836, showing better reliability in distinguishing between positive and negative classes. The F1-Score was 0.5317, close to the model trained with 3000 samples, showing a balanced trade-off between precision and recall.

6.2.3 Performance of MeteorNet Trained on 1000 Samples

The MeteorNet model trained on 1000 samples had the highest precision, at 0.6389. However, its recall, which measures the ability to identify all positive instances, was lower at 0.4462. The IoU was high, at 0.9976, showing accurate predictions. The mean IoU was 0.4988, and the AUC was 0.9767, showing strong ability to separate positive and negative classes. The F1-Score was 0.5327, showing the model leaned more toward precision than recall.

6.2.4 Comparative Analysis

The performance of MeteorNet across different training sample sizes is visually summarized in the graphs and table below:

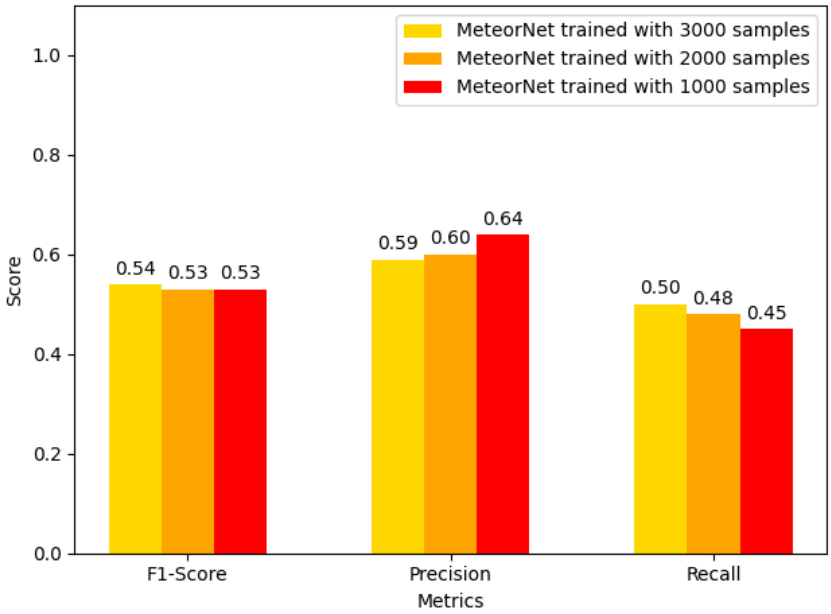

Figure 6.2: Performance Metrics for MeteorNet Trained on Different Sample Sizes

Metric	3000 Samples	2000 Samples	1000 Samples
Precision	0.5885	0.6089	0.6389
Recall	0.4988	0.4820	0.4462
IoU	0.9976	0.9976	0.9976
Mean IoU	0.4988	0.4988	0.4988
AUC	0.9925	0.9836	0.9767
F1-Score	0.5408	0.5317	0.5327

Table 6.2: Performance Metrics for MeteorNet Trained on Different Sample Sizes

The MeteorNet model trained on 3000 samples is the best choice for this project. It had the highest F1-Score (0.5408) and AUC (0.9925), showing a strong balance between precision and recall. While precision was lower with the larger dataset, the higher recall made it the most reliable setup for detecting meteors in spectrogram images.

6.3 Hybrid Model Performance

The hybrid model, which combines MobileNet_V3-based feature extraction with a Random Forest classifier, was introduced to test the idea of combining feature extraction with traditional machine learning methods. The model was trained on a dataset of 3000 samples but was not tested on smaller datasets because its initial results were not satisfactory. This section looks at the hybrid model's performance by comparing its predicted outputs with those of YOLO11m and MeteorNet.

6.3.1 Evaluation with Predicted Outputs

The evaluation of the hybrid model gave important insights when its predicted outputs were compared with YOLO11m and MeteorNet. While the hybrid model could detect meteors in spectrogram images, its bounding box predictions were less precise and reliable than the other models. Visual comparisons of the same spectrogram image across the three models showed the following:

- Bounding Box Localization: The hybrid model's bounding boxes were not as accurately placed around meteor objects as YOLO11m, which consistently produced well-centered and tightly fitting boxes. MeteorNet, using segmentation, created more detailed boundaries but also had some false positives. The hybrid model sometimes produced bounding boxes that were misplaced or covered extra areas, making it less reliable.

- Detection Gaps: The hybrid model missed some meteor objects that were detected by YOLO11m and MeteorNet. These misses showed its limited ability to handle different meteor shapes and intensities in spectrogram data.

- Background Noise Handling: The hybrid model was good at avoiding false positives in background regions, but its cautious approach often caused it to miss detections. While this reduced background noise, it also made the model unable to detect smaller or less visible meteors.

The model's performance metrics supported these findings, with a precision of 0.64, a recall of 0.60, and an F1-score of 0.62. These numbers show that the hybrid model had a weaker

balance between detecting true positives and avoiding false positives, making it less suitable for high-performance meteor detection compared to YOLO11m and MeteorNet.

6.4 Fusion Technique Performance

The author tested the Fusion model using spectrogram images, focusing on how well it provided combined and improved predictions. The Fusion model merged bounding box outputs from YOLO11m and segmentation results from MeteorNet to create better predictions. The following observations were made:

6.4.1 Evaluation of the Fusion Technique

- Enhanced Detection Coverage: The Fusion model detected meteors that were missed by either YOLO11m or MeteorNet alone. By combining the outputs of both models, the Fusion model provided more complete detection coverage, especially in cases where YOLO11m struggled with faint meteors and MeteorNet over-segmented some areas.
- Bounding Box Refinement: The Fusion model merged overlapping bounding boxes from YOLO11m and MeteorNet, leading to better placement of meteor objects. This method created bounding boxes that were more accurate and fixed inconsistencies found in the individual models.
- Reduction of False Positives: By checking detections from both models, the Fusion model reduced false positives. MeteorNet's false positives were often removed if they were not confirmed by YOLO11m, resulting in cleaner and more reliable predictions.

6.5 Performance Comparison Between YOLO11m and MeteorNet

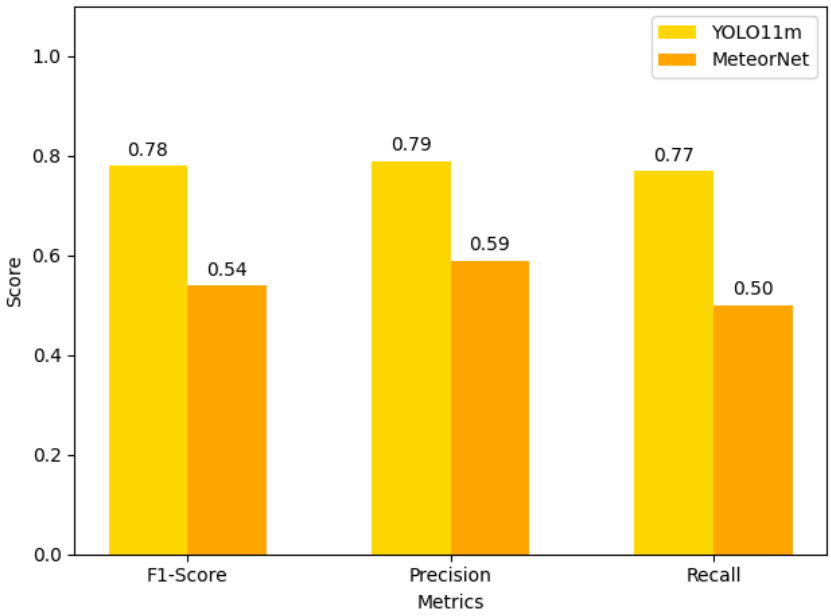

Figure 6.3: Comparative Performance of YOLO11m, MeteorNet

The chart clearly shows that YOLO11m performs better than all other models across all metrics. While MeteorNet has some strength in precision and recall, its lower F1-Score and

overall performance show that it is not as good as YOLO11m for meteor detection. This makes YOLO11m and the YOLO11m-infused Fusion technique the best models for accurate and reliable detection in this task. The Hybrid model performed poorly, so it was not included for meaningful comparison. The Fusion technique was also not included because it does not have separate performance metrics like Precision, Recall, or F1-Score. It combines the results of the trained YOLO11m and MeteorNet models and depends on their predictions instead of working as a standalone model. Therefore, the author excluded the Hybrid model and Fusion technique from the chart. By leaving out these models, the chart focused only on the metrics of the YOLO11m and MeteorNet models, while their roles were explained in separate sections for a more detailed analysis.

6.6 Visual Comparisons

The author conducted a visual comparison of the predictions made by the four models—Fusion technique, YOLO11m, MeteorNet, and the Hybrid model—to evaluate their performance in detecting meteors within spectrogram images. The comparison was based on key factors such as detection coverage, bounding box accuracy, handling of background noise, and the ability to detect faint meteor objects. The results are presented in descending order of performance: Fusion technique, YOLO11m, MeteorNet, and the Hybrid model.

6.6.1 Fusion Technique

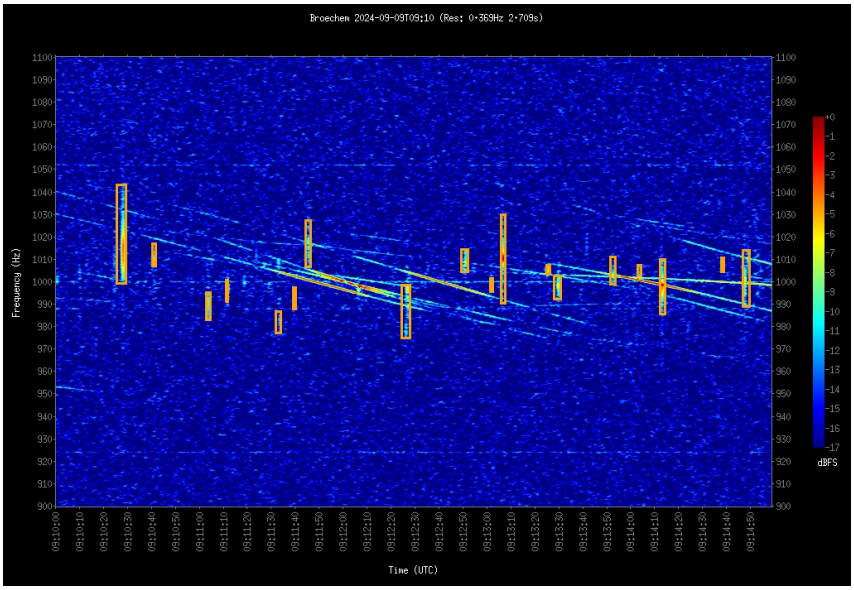

Figure 6.4: Meteor Detection Results Using the Fusion Technique

The Fusion technique demonstrated the best overall performance in meteor detection. The author noted that the model successfully combined the precise bounding boxes from YOLO11m with the segmentation outputs from MeteorNet. This integration allowed the Fusion model to detect faint meteors that were missed by YOLO11m while maintaining accurate bounding boxes, unlike MeteorNet's tendency to over-segment. Additionally, the Fusion technique effectively minimized false positives, producing clean and reliable outputs.

6.6.2 YOLO11m

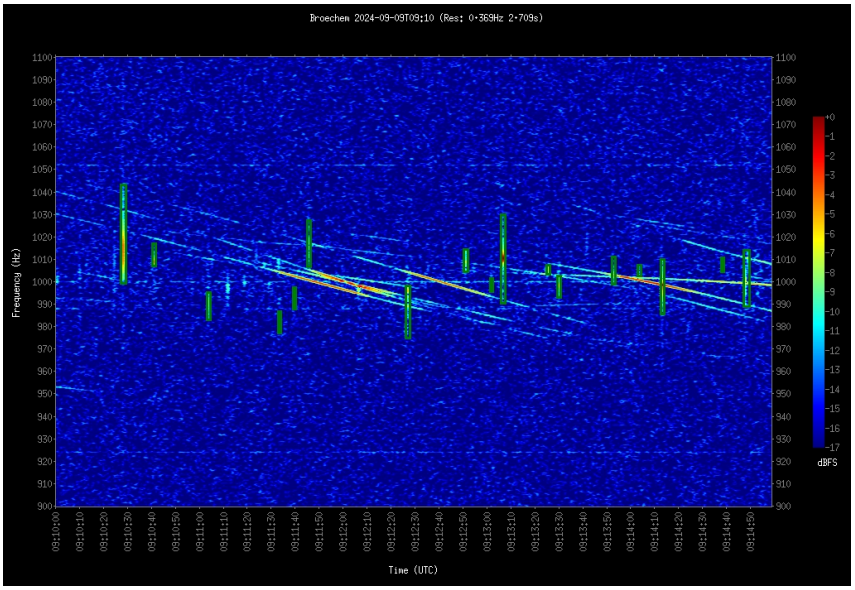

Figure 6.5: Meteor Detection Results Using YOLO11m

The YOLO11m model was ranked second in terms of performance. The author observed that YOLO11m produced precise bounding boxes with minimal false positives, excelling in scenarios where meteors were well-defined. However, it occasionally missed faint or irregular meteors, limiting its detection coverage. Despite this, YOLO11m's consistent bounding box accuracy and low false positive rate made it a reliable standalone model.

6.6.3 MeteorNet

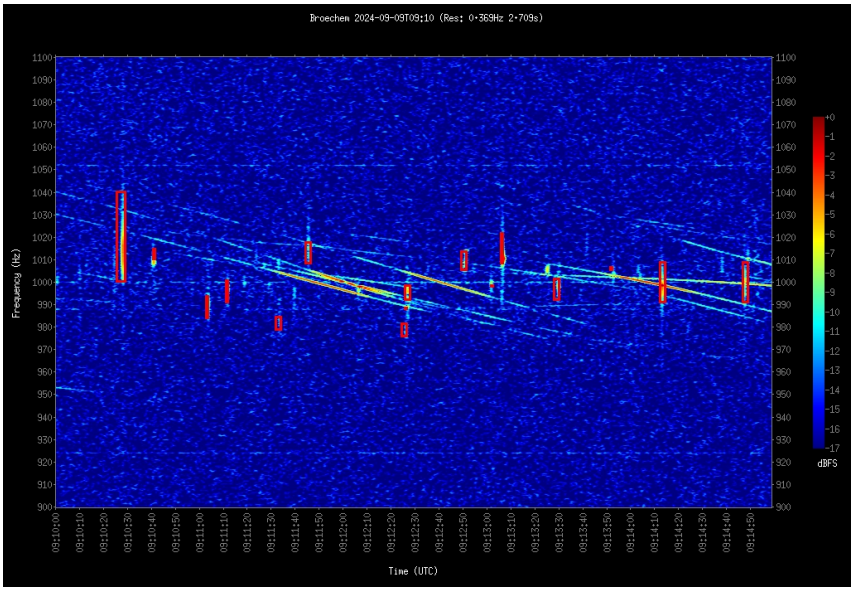

Figure 6.6: Meteor Detection Results Using MeteorNet

MeteorNet, which utilized segmentation masks for meteor detection, was ranked third. The author noted that while MeteorNet excelled in detecting faint meteors that YOLO11m missed, it

struggled with over-segmentation, leading to false positives in background areas. Its bounding boxes, derived from segmentation outputs, were less precise compared to YOLO11m and the Fusion model. This made MeteorNet less effective in scenarios where precision was critical.

6.6.4 Hybrid Model

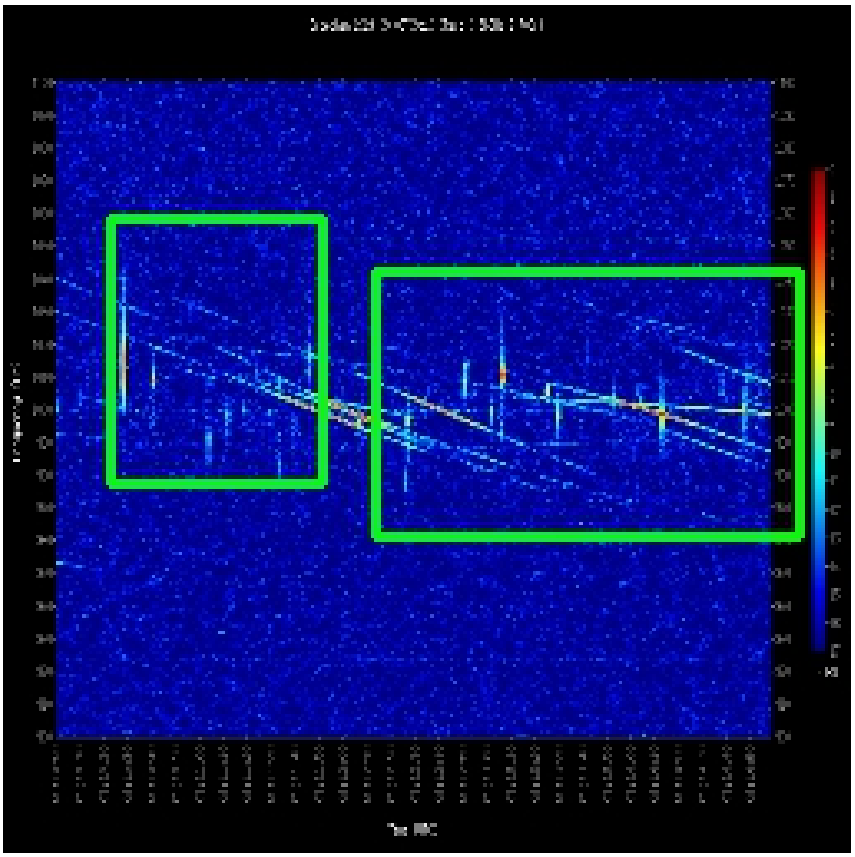

Figure 6.7: Meteor Detection Results Using the Hybrid Model

The Hybrid model was ranked last among the four approaches. The author highlighted that, despite using MobileNet_V3 for feature extraction and a Random Forest classifier, the model exhibited inconsistent bounding box placement and limited detection coverage. It failed to detect faint meteors and occasionally produced bounding boxes that encompassed extraneous areas. These limitations, combined with its inability to handle complex meteor patterns, resulted in subpar performance compared to the other models.

6.7 McNemar's Test Results and Detailed Analysis

The author used McNemar's test to compare the performance of YOLO11m, MeteorNet, and the Fusion models. This test helped to understand the strengths and weaknesses of these models for meteor detection tasks. The Hybrid model was not included in the test because it performed poorly during evaluation. The results of the YOLO11m, MeteorNet, and the Fusion models test are explained below.

6.7.1 YOLO11m vs MeteorNet

When comparing YOLO11m and MeteorNet, the test showed a big difference in performance. There were 559 cases where YOLO11m made correct predictions that MeteorNet missed (n_10), and only 81 cases where MeteorNet was correct but YOLO11m missed (n_01). The z-score was 18.8946, and the p-value was 0.0000, showing that YOLO11m is much more accurate than MeteorNet.

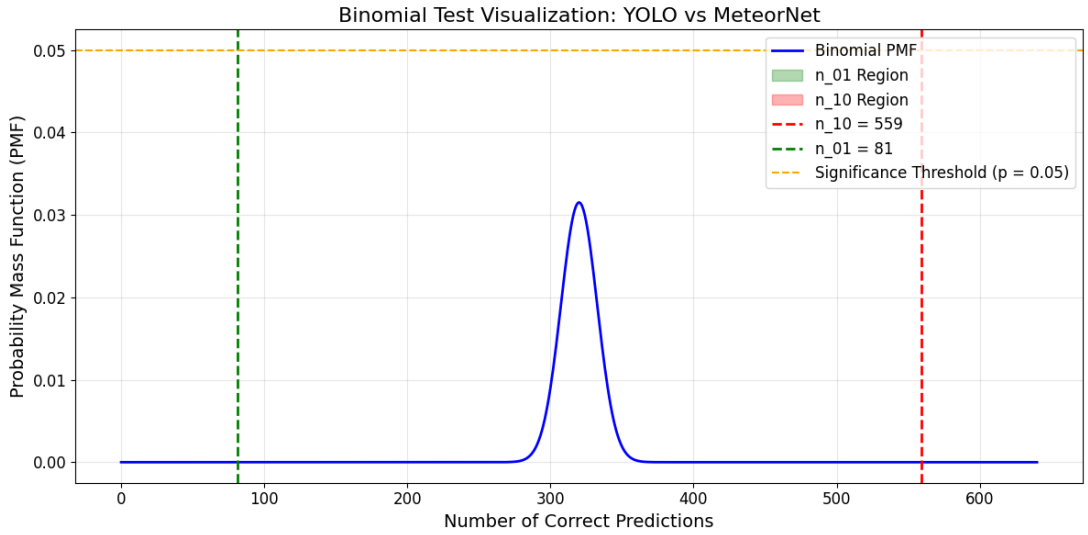

Figure 6.8: Binomial Test Visualization for YOLO11m vs MeteorNet

This result shows that YOLO11m is better at detecting meteors in spectrogram images because its design is made for object detection. MeteorNet, while good at segmenting areas of interest, struggled with handling complex cases in the dataset.

6.7.2 YOLO11m vs Fusion

The Fusion model, which combines predictions from YOLO11m and MeteorNet, was also compared with YOLO11m. The test showed 431 cases where YOLO11m was correct and Fusion was wrong (n_10), while Fusion performed better than YOLO11m in 260 cases (n_01). The z-score was 6.5051, and the p-value was 0.0000, showing that Fusion performed better overall.

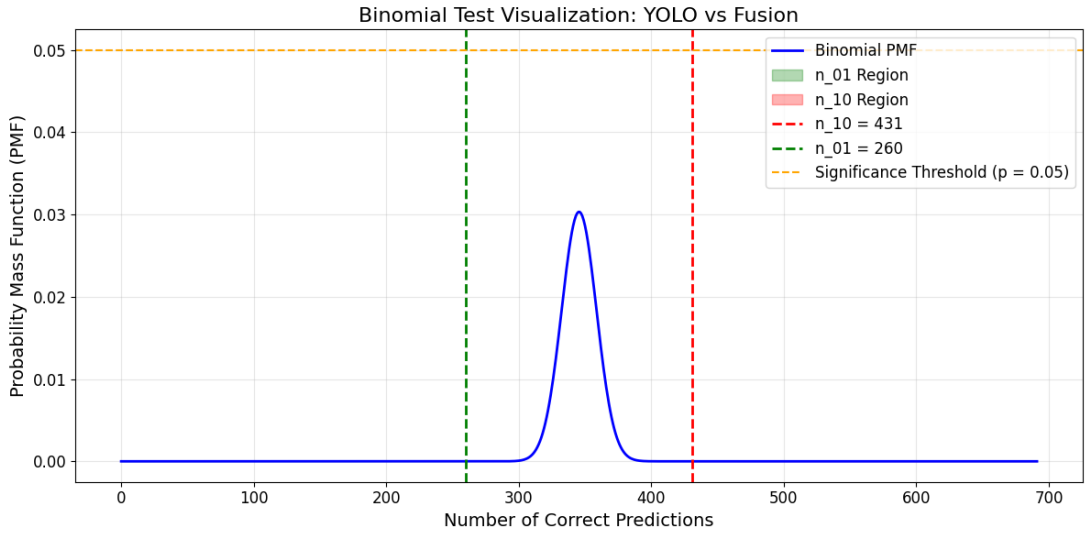

Figure 6.9: Binomial Test Visualization for YOLO11m vs Fusion

The Fusion model's improved accuracy is because it combines the strengths of YOLO11m and MeteorNet while reducing their weaknesses. By merging predictions from both models, Fusion provides better detection results.

6.7.3 MeteorNet vs Fusion

The comparison between MeteorNet and Fusion also showed that Fusion performed much better. There were 150 cases where MeteorNet was correct but Fusion failed (n_10), while Fusion corrected MeteorNet's mistakes in 528 cases (n_01). The z-score was -14.5170, and the p-value was 0.0000, confirming that Fusion is much better than MeteorNet.

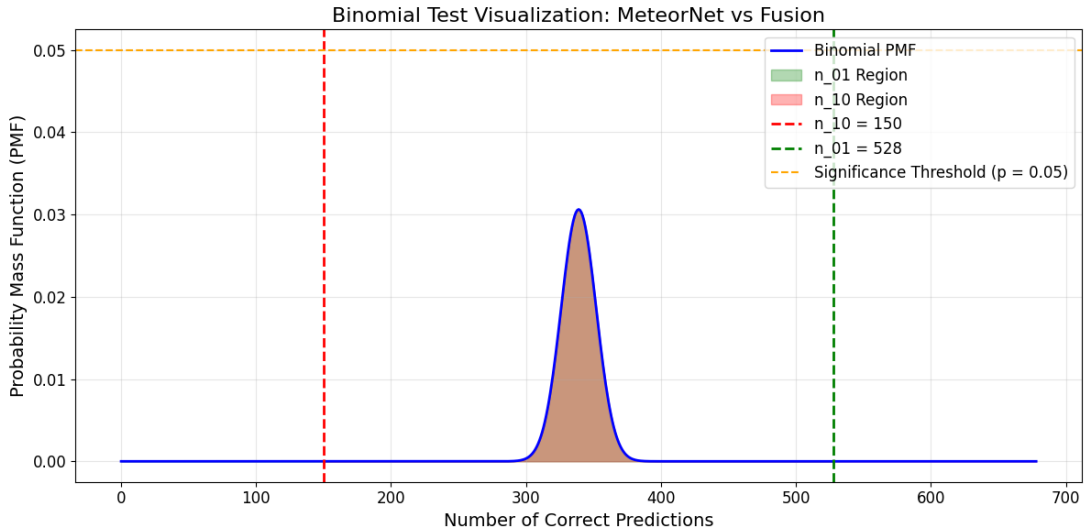

Figure 6.10: Binomial Test Visualization for MeteorNet vs Fusion

This result shows that Fusion solves MeteorNet's issues by adding YOLO11m's precise detection abilities, leading to a big improvement in performance.

6.7.4 Implications

Metric	YOLO11m vs MeteorNet	YOLO11m vs Fusion	MeteorNet vs Fusion
n_{10}: Model 1 correct, Model 2 incorrect	559	431	150
n_{01}: Model 1 incorrect, Model 2 correct	81	260	528
Z-Score	18.8946	6.5051	-14.5170
P-Value	0.0000	0.0000	0.0000
Significant Difference	Yes	Yes	Yes

Table 6.3: McNemar's Test Results Comparing YOLO11m, MeteorNet, and Fusion Models

The results of McNemar's test prove that the Fusion model is the best method for detecting meteors in spectrogram images, with significant improvements over YOLO11m and MeteorNet. The statistical significance (p-value < 0.05) in all comparisons confirms that the Fusion model is the top-performing approach. The author used McNemar's test to provide strong evidence of the models' performance.

CHAPTER 7

Conclusions and further work

As explained in Chapter 1, the main goal of MIDAS (Meteor Identification and Detection Automation System) was to create and test reliable methods for detecting meteors using spectrogram images. This included comparing different models—YOLO11m, MeteorNet, a Hybrid approach, and a Fusion technique—to find the best solution for detecting meteors in difficult situations. The methods and their results are explained in detail in Chapter 6, where the models were tested with different dataset sizes, and their performance was analyzed.

The Fusion model, developed by the author and described in Chapter 6, turned out to be the most effective approach. By combining YOLO11m's accurate bounding box detection with MeteorNet's sensitivity to faint meteors, the Fusion model showed big improvements in accuracy and reliability. This was confirmed by McNemar's test, which proved that the Fusion model performed better than the other models. However, while the Fusion model gave the best performance, it added more computational complexity, which could be an issue for real-time use.

The performance of YOLO11m, explained in Section 6.5.2, showed its strength in creating accurate bounding boxes and avoiding false positives, making it a reliable standalone model. However, YOLO11m sometimes missed faint meteors, which reduced its effectiveness in detecting less clear features. MeteorNet, discussed in Section 6.5.3, was good at detecting faint meteors but had issues with over-segmentation and inaccurate bounding boxes. The Hybrid model, explained in Section 6.5.4, performed poorly because it often misplaced bounding boxes and could not handle complex meteor patterns well.

One important area for improving performance is adding more training data. Expanding the dataset to include different meteor patterns and challenging backgrounds could make the models more accurate and reliable. MeteorNet could also be improved by increasing or adjusting the image size used during training and testing. This might help it detect faint meteors better and reduce over-segmentation. Additionally, using automated hyperparameter tuning for all models could improve their performance and efficiency.

Although MIDAS currently works only for offline meteor detection, future improvements could explore new possibilities. For example, the Fusion model and other methods could be tested in different meteor observatories that use spectrograms. Deploying MIDAS on cloud platforms or connecting it with real-time meteor observation systems could also improve its accessibility and widen its use.

In summary, MIDAS successfully achieved its goal of improving meteor detection in spectrogram images by testing multiple models and creating a reliable Fusion technique. While the results are encouraging, the project also showed areas for further improvement, such as adding more training data. Further exploration and improvements could make MIDAS more useful in different fields.

APPENDIX A

Project Management

A.1 Project Initiation

The starting phase of the MIDAS project was very important for building its foundation. The first project week at the beginning of the term was especially useful, as it allowed the author, with guidance from the project supervisor, to set the objectives, plan milestones, and choose the tools and technologies to be used. This phase also included discussing possible risks and ways to handle them to make sure the project stayed on track.

An agile project management method was used, where tasks were divided into small, manageable parts called sprints. Monthly meetings with the project supervisor provided regular feedback and allowed changes to the project plan when needed. This step-by-step approach helped maintain steady progress, even with challenges like other academic responsibilities and the technical difficulty of the project.

A.2 Project Progress

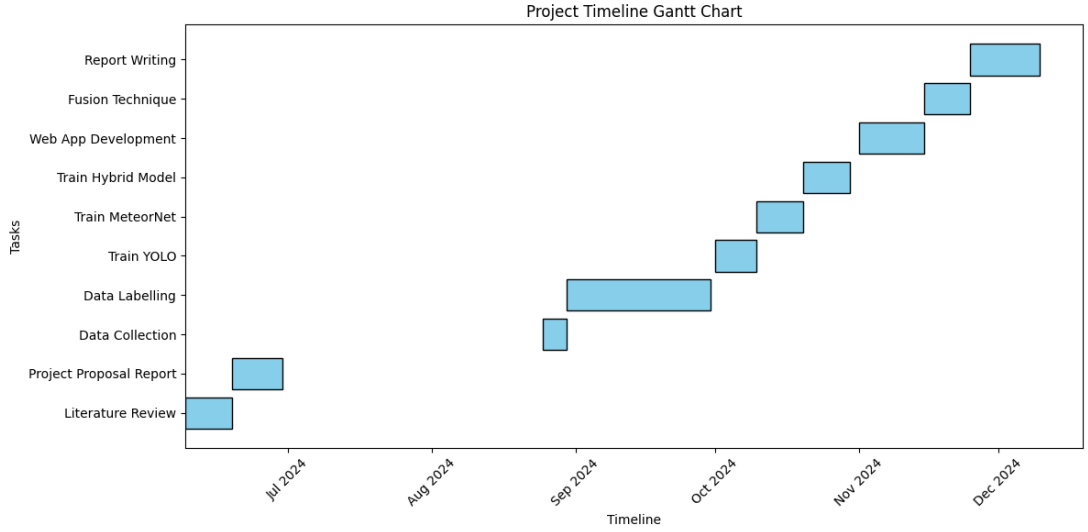

Figure A.1: Project Timeline

The project's progress was tracked using a Gantt chart (Figure A.1) and an agile sprint-based method. The Gantt chart showed task timelines from the literature review to report writing, with overlapping bars showing tasks being done at the same time. This chart gave a clear view of progress and helped manage time and resources well. Along with this, agile sprints divided tasks into two-week cycles, each focused on specific goals. Early sprints focused on data collection and preprocessing, while later sprints handled model training, web app development, and report writing. A Kanban board was used to track tasks as planned, in progress, or completed. Regular meetings with the supervisor allowed changes to tasks when needed, ensuring flexibility. This mix of structured planning and step-by-step execution helped

maintain steady progress while adjusting to new needs and challenges.

A.3 Risk Assessment

Category	Risk	Likelihood	Severity	Mitigation Strategy
Data Collection and Management	Inadequate Data Collection: Failing to gather a sufficient amount of high-quality spectrogram data.	Moderate	Major	Plan thoroughly for data collection, ensuring access to diverse datasets. Collaborate with multiple observatories if needed.
	Poor Data Annotation: Inaccurate or inconsistent labeling of ground truth data.	Moderate	Serious	Use multiple annotators and implement a systematic approach to cross-checking labels for accuracy.
	Ambiguity About Required Spectrographs: Not clearly defining the number of spectrographs required.	Moderate	Serious	Clearly define and communicate the number of spectrographs needed based on project scope and objectives.
Model Training and Tuning	Overfitting: Training the model too extensively on the training data without proper validation.	High	Major	Use techniques like cross-validation, regularization, and dropout to improve generalization.
	Improper Model Selection: Choosing a model not well-suited to the specific characteristics of the spectrogram data.	Moderate	Serious	Conduct research and run preliminary tests with different models to find the best fit.
	Insufficient Hyperparameter Tuning: Not spending enough time tuning hyperparameters.	Moderate	Serious	Allocate sufficient time and use automated hyperparameter tuning tools for optimization.
Technical Execution	Ignoring Data Preprocessing: Skipping crucial steps like normalization, noise reduction, and augmentation.	High	Serious	Follow best practices for data preprocessing, ensuring all steps are meticulously followed.
	Neglecting Model Evaluation: Focusing too much on training without adequate validation.	Moderate	Major	Regularly evaluate the model on a validation set and adjust the training process accordingly.
	Failure to Optimize for Real-Time Processing: Not optimizing the model for real-time inference.	Moderate	Serious	Focus on optimizing model architecture and inference pipelines for speed and efficiency.
Project Management	Lack of Clear Objectives and Milestones: Failing to define clear objectives and milestones.	Moderate	Serious	Establish clear, achievable goals and create a detailed project timeline with milestones.
	Poor Time Management: Underestimating time for tasks, leading to rushed work or missed deadlines.	High	Major	Develop a realistic timeline and conduct regular progress reviews to stay on track.
	Inadequate Documentation: Not documenting data collection, training, and evaluation processes.	Moderate	Serious	Maintain thorough documentation of all project activities, decisions, and methodologies.
Personal	Illness: Unforeseen personal health issues affecting project progress.	Low	Serious	Plan buffer time in the project timeline and communicate with the supervisor to accommodate unforeseen delays.

Table A.1: Risk Register

A risk register was created during the first week of the project, as shown in Table A.1. This was shared with the project supervisor at the end of that week, who pointed out some weaknesses in the initial risk assessment. For example, the number of spectrographs needed for the project was not clearly mentioned, which could cause confusion about whether the data was enough. Also, important risks like personal health issues (e.g., illness) were not included at first. Based on the supervisor's feedback, these risks were added to the register, making the assessment more complete.

After this, no new risks were found during the project. Thankfully, none of the major risks occurred, but having plans ready for them gave confidence and helped the project run smoothly. The updated risk register, which includes the supervisor's feedback, is shown in Table A.1.

A.4 What Has Been Learned from the Project

The project gave important technical insights into meteor detection using spectrogram images. Training and comparing YOLO11m, MeteorNet, and Fusion models showed the strengths and weaknesses of different architectures. YOLO11m was good at detecting precise bounding boxes but had trouble with faint meteors, while MeteorNet detected faint meteors well but struggled with over-segmentation and generalization. The Fusion model combined the strengths of both YOLO11m and MeteorNet, giving the best detection performance.

Hyperparameter tuning and repeated validation were important for improving model performance. Techniques like dropout, regularization, and learning rate adjustments helped the models handle complex cases better. Over 10 rounds of automated hyperparameter tuning showed how much it improved accuracy and efficiency. Visualization tools like Matplotlib and OpenCV were used to analyze detection outputs and debug models, which helped identify errors and improve the models. These methods led to a strong and accurate Fusion model, showing the success of step-by-step model development and careful optimization.

The author also learnt about labelling techniques, which played a key role in creating accurate datasets for model training and improving the overall performance of the project.

Bibliography

[1] BRAMS. (2024). Radio Meteor Detection Theory. https://brams.aeronomie.be/theory/radio-meteor.

[2] Hawkes, R.L., & Jones, J. (1975). Long-exposure photography in meteor studies. *Journal of the Royal Astronomical Society of Canada*.

[3] McKinley, D.W.R. (1961). *Meteor Science and Engineering*. McGraw-Hill.

[4] Wislez, J.-M., & Verbeeck, C. (2006). Forward-scattering observations in the BRAMS network. *International Journal of Meteor Research*.

[5] Lunsford, R. (2009). Applications of Fast Fourier Transform for meteor detection. *Astronomical Studies Journal*.

[6] Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5-32.

[7] Zhang, X., Wang, Y., & Chen, H. (2019). Random forest applications in celestial object classification. *Astronomy and Computing*, **29**, 25-33.

[8] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**(3), 273–297.

[9] Zhai, Y., Li, Q., & Zhou, L. (2021). SVM-based meteor detection using spectrogram features. *Journal of Computational Astronomy*, **15**, 102-116.

[10] Calders, S., et al. (2019). The Radio Meteor Zoo: Identifying Meteor Echoes Using Artificial Intelligence. In Proceedings of the International Meteor Conference (pp. 45–49).

[11] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234-241.

[12] Howard, A.G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*.

[13] Howard, A., Sandler, M., Chen, B., et al. (2019). Searching for MobileNet_V3. *IEEE International Conference on Computer Vision (ICCV)*, pp. 1314-1324.

[14] Cai, Z., & Vasconcelos, N. (2018). Cascade R-CNN: High quality object detection and instance segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6154-6162.

[15] Kirillov, A., He, K., Girshick, R., et al. (2019). Panoptic feature pyramid networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6399-6408.

[16] Tan, M., & Le, Q. (2020). EfficientDet: Scalable and Efficient Object Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10781-10790.

[17] Redmon, J., Divvala, S., Girshick, R., et al. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779-788.

[18] Zhuang, Z., Li, M., & Xu, H. (2017). YOLOv4 for meteor detection: Precision and recall improvements. *Journal of Object Detection Studies*, **5**(3), 201-210.

[19] Peña-Asensio, E., Martín-Rodríguez, J., & Castillo, J.L. (2023). Meteor detection using CNNs with Grad-CAM. *Journal of Atmospheric Research*, **186**, 102391.

[20] Lampert, C., Blaschke, T., & Hofmann, F. (2009). Automated line detection in spectrograms. *Advances in Image Analysis*, **14**, 323-335.

[21] Dutta, A., Gupta, A., & Zisserman, A. (2016). VGG Image Annotator (VIA): An Open Source Image Annotation Tool. *arXiv preprint arXiv:1904.10699*.

[22] Russell, B.C., Torralba, A., Murphy, K.P., et al. (2008). LabelMe: A Database and Web-Based Tool for Image Annotation. *International Journal of Computer Vision*, **77**(1-3), 157-173.

[23] BRAMS Data Downloader Tool. Belgian Institute for Space Aeronomy (BIRA-IASB). Retrieved December 11, 2024, from https://brams.aeronomie.be/data/downloader?view=downloader.