Tree Detection from Satellite Images via Merged CNN Models

Submitted in partial fulfillment of the
requirements for the degree

of

Bachelor of Science in Electrical and Electronics Engineering

by

İbrahim Tınas 150719046

Under the supervision of

Prof. Cem Ünsalan

MARMARA
UNIVERSITY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

FACULTY OF ENGINEERING

MARMARA UNIVERSITY

September, 2024

DECLARATION OF AUTHORSHIP

Project Title	Tree Detection from Satellite Images via Merged CNN Models
Authors	*İbrahim Tınas*
Supervisor	Prof. Cem Ünsalan

İbrahim Tınas hereby declare that this project titled *Tree Detection from Satellite Images via Merged CNN Models* is entirely our own work, unless otherwise referenced or acknowledged. The content of this project is the result of our own research and efforts, and we have complied with all ethical guidelines and academic standards.

We are aware of the consequences of academic dishonesty and understand that any violation of ethical standards in this project may lead to disciplinary actions as defined by Marmara University's policies.

İbrahim Tınas
150719046

Istanbul
September, 2024

MARMARA
UNIVERSITY

Electrical and Electronics Engineering
Engineering
Marmara University
Istanbul, Türkiye

Tree Detection from Satellite Images via Merged CNN Models

by

İbrahim Tınas

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

The author(s) hereby grant(s) to Marmara University permission to reproduce and to distribute publicly paper and electronic copies of this document in whole or in part and declare that the prepared document does not in any way include copying of previous work on the subject or the use of ideas, concepts, words, or structures regarding the subject without appropriate acknowledgement of the source material.

Signature of Author(s)

Supervisor
Prof. Cem Ünsalan

Head of Department
Prof. Gökhan Bora Esmer

ACKNOWLEDGEMENTS

To begin with, I want to extend my sincerest thanks to Prof. Cem Ünsalan for his consistent support and guidance during the running of this project.
I am also grateful to my classmates and family members for motivating me.

İbrahim Tınas
Marmara University
Date: September, 2024

ABSTRACT

This research aims to enhance the accuracy and efficiency of Faster R-CNN, optimized for tree detection in aerial photographs, by employing a dual-backbone architecture combining MobileNetV3 and ResNet101. The study evaluates the performance of this approach on multiple datasets, including the Oil Palm Tree Detection Dataset, Turkey Dataset, and Katam Dataset. The dual-backbone model demonstrates significant improvements in detection accuracy and computational efficiency, achieving high mean Average Precision (mAP) scores across various thresholds. This research highlights the potential of combining lightweight and deep network architectures for practical applications in environmental monitoring and sustainable forest management. The findings underscore the importance of advanced deep learning techniques in promoting ecological balance and forest conservation.

Keywords: Aerial Imagery, Artificial Intelligence, Convolutional Neural Networks, Deep Learning, Faster R-CNN, MobileNetV3, ResNet101, Tree Detection, Dual-Backbone Architecture

ABBREVIATIONS

TÜİK	Turkish Statistical Institute
CNN	Convolutional Neural Network
R-CNN	Region Based Convolutional Neural Networks
ReLu	Rectified Linear Unit
FC	Fully Connected
RPN	Region Proposal Network
IoU	Intersection over Union
AP	Average Precision
mAP	Mean Average Precision
TP	True Positive
FP	False Positive
FN	False Negative
ODS	Object Detection System
GPU	Graphics Processing Unit
SGD	Stochastic Gradient Descent
FPN	Feature Proposal Network
ROI	Region of Interest
COCO	Common Objects in Context

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Thesis Background

As of 2023, Turkey's total forest area is approximately 23.245 million hectares, accounting for 29.8% of the country's total land area. According to the Turkish Statistical Institute, Antalya province stands out as the region with the largest forest area in Turkey, with over 1 million hectares of forest land. The regions with the most extensive forest areas in Turkey are the Black Sea, Mediterranean, Aegean, and Central Anatolia regions, respectively.

In light of these extensive forest areas, the use of deep learning techniques in aerial image processing, specifically Convolutional Neural Networks, has become increasingly significant. This study focuses on the efficiency of Faster R-CNN in detecting areas of interest in aerial images, particularly for tree detection. Researchers have employed advanced neural networks to enhance the accuracy of these deep learning methods, demonstrating their potential in this domain.

Addressing ecological problems requires the application of advanced deep learning techniques in tree detection. These efforts contribute to global initiatives for sustainable forest management. Advanced tree detection methods play a crucial role in mitigating climate change, preventing erosion, and protecting the environment. Therefore, improving these techniques is of paramount importance.

However, traditional tree detection methods often involve a trade-off between accuracy and computational cost. Deep learning models offer a promising alternative, as they can produce more accurate results across different tree types and image resolutions. Yet, they usually come with higher computational costs. To address this, our study merges the strengths of MobileNetV3 and ResNet101 as backbones in the Faster R-CNN framework. This approach aims to balance computational efficiency and accuracy, leveraging

MobileNetV3's lower computational cost and ResNet101's high accuracy to enhance environmental monitoring applications.

Building on existing research, this study aims to improve the accuracy of aerial image analysis for tree detection. By integrating the merged backbones of MobileNetV3 and ResNet101 in the Faster R-CNN framework, we seek to overcome current challenges in the field. These developments are essential for enhancing resource efficiency and effectiveness in large-scale environmental monitoring applications.

1.2 Project Definition

1.2.1 Problem Statement

This study aims to create an optimized Faster R-CNN model with a combination of MobileNetV3 and ResNet101 backbones for more efficient and accurate tree detection in aerial images. Specifically, in terms of accuracy and processing efficiency, this study explores ways to surpass the limitations of current tree detection methodologies.

1.2.2 Context and Scope

The scope of this study includes developing and validating the enhanced Faster R-CNN model, focusing on tree detection in diverse aerial images. Two different datasets are utilized in this study to evaluate the tree detection model. Each dataset presents unique challenges and features to ensure comprehensive and robust model training. The datasets are described as follows:

- **Turkey Dataset:** This dataset comprises 1,468 images from various locations in Turkey, including Bursa, Adana, Gaziantep, and Antalya, collected via Google Earth. The images encompass a range of tree sizes and include both natural and man-made forests to ensure diversity. Additionally, the dataset incorporates various soil colors and natural geographical features to test the model's capacity to detect trees under different background conditions. In total, this dataset contains 86,971 labels, intended to train the model to recognize various tree species.
- **Oil Palm Tree Detection Dataset:** Collected from Roboflow, this dataset is designed for oil palm tree detection and consists of 922 images with 9,874 labels from aerial imagery, contributing to resilient model training.

1.2.3 Significance and Implications

Integrating advanced tree detection methods is essential for sustainable forest management. Although forests cover approximately 31% of the global land area, deforestation remains a significant problem. According to the FAO, net forest loss fell from 7.8 million hectares per year in the 1990s to 4.7 million hectares yearly in the 2010s. However, more than this reduction rate is needed to prevent deforestation completely. Primary forests, essential for biodiversity and carbon storage, are still rapidly depleted. The enhanced efficiency and accuracy will be useful in different environmental work, from observing deforestation over time to helping to protect natural habitats for endangered animals.[1]

Chapter 2

RESEARCH OBJECTIVE

When advanced machine learning methods are applied, especially in aerial image analysis, new paths of environmental monitoring and resource management can be discovered. This chapter outlines the main goal of the study, which is to develop an advanced and sustainable tree detection model using aerial imagery. This detection model aims to significantly enhance the accuracy and processing speeds of previous tree detection techniques.

This research builds upon combining the enhanced qualities of the Faster R-CNN structure with the efficiency of MobileNetV3 and ResNet101 backbones. The combination is expected to create a much stronger and more efficient model that can analyze various aerial images quickly and accurately. The goals of this research include both theoretical insights and quantifiable metrics.

The following main objectives form the backbone of this research:

- **Objective 1: Development of a Tree Detection Model**

  Develop a tree detection model using aerial image datasets, integrating the Faster R-CNN with both MobileNetV3 and ResNet101 backbones to enhance computational efficiency and accuracy for various images.

- **Objective 2: Improvement in Detection Accuracy**

  Design a structure that significantly improves the detection accuracy of trees across different geographical formations. Achieve a notable increase in the mAP@50:95 score, surpassing current detection methods.

- **Objective 3: Model Adaptability Across Diverse Landscapes**

  Ensure the model's adaptability and effectiveness among different types of trees and tree densities. Validate the model's performance across two different datasets representing various geographical variations and tree densities.

- **Objective 4: Optimization of Model Training and Evaluation Process**

  Simplify and enhance the model's training process by improving training techniques, hyperparameters, and data transformations strategies.

- **Objective 5: Comparative Analysis of Backbones**

  Conduct a comparative analysis of the results using MobileNetV3, ResNet101, and the combination of MobileNetV3 and ResNet101 as backbones. Evaluate the performance differences in terms of accuracy and memory usage.

2.1 Project Specifications

2.1.1 Convolutional Neural Networks

2.1.1.1 Convolution Operation:

CNNs are deep neural networks precisely prepared for processing grid-like data, such as images and video. An input is processed to produce an output feature map in a CNN by applying a filter, also called a kernel, through a convolution operation.

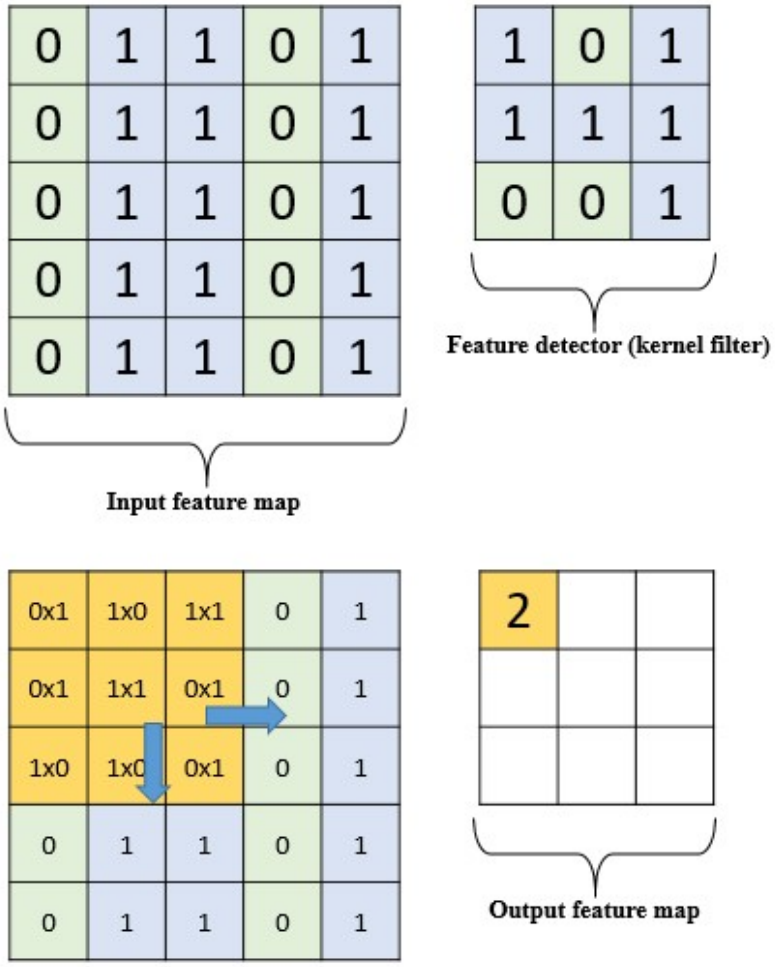


Figure 2.1: Convolutional operation.

As shown in Figure 2.1, the kernel filter, which has the size of the defined padding value, moves around the entire image pixel by pixel according to the defined striding value and applies the operation defined in the filter. The purpose of this process is to produce more

accurate feature maps by reducing the size of the image without losing features, as seen in the figure.

2.1.1.2 Convolution Layers:

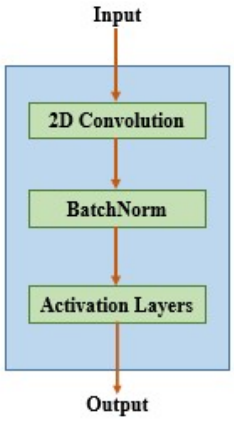


Figure 2.2: Conv2dNormActivation layer.

Conv2dNormActivation is likely a custom or framework-specific layer that combines convolution (Conv2D), normalization (such as batch normalization), and activation (like ReLU or Swish) in a single layer, as shown in Figure 2.2. It streamlines the application of These operations, which are typically performed sequentially in a CNN.

2.1.1.2.1 2D Convolution Layers

Convolution layers (Conv2D) apply the convolution operation as depicted in Figure 2.1 to the input. Each Conv2D layer can extract different features by learning multiple filters. The layer's output, a feature map or activation map, represents the presence of learned features in the input.

2.1.1.2.2 Normalization Layers

- **Batch Normalization**

  As shown in Figure 2.2, batch normalization plays a crucial role in normalizing the output of a layer in the hidden layer. This normalization benefits training time and convergence speed. Due to the more standardized layer inputs, the model does not encounter unexpected data as before, and model learning rates can be defined at much higher values.

2.1.1.2.3 Activation Layers

- **Rectified Linear Unit (ReLU):**

  ReLu functions are used in activation layers, as shown in Figure 2.2. If the Relu input value exceeds 0, it gives the same value. If the input value is less than 0, it assigns the output value to 0 as depicted in Figure A.4. The feature maps created in CNN are multiplied and summed by the weight of the kernel filter, as seen in Figure2.1. This feature map, to which the ReLu activation function is applied, will eliminate values less than 0. In this case, it will reduce CNN's training time and cause it to learn in a shorter time,

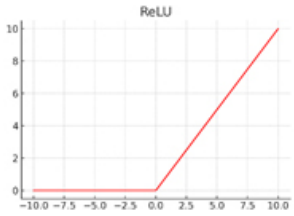


Figure 2.3: Graph of ReLU.

- **Hard Sigmoid:**

  Hard sigmoid can be preferred in hardware-friendly artificial intelligence applications and devices with limited resources. Since it does not have a vanishing gradient problem, unlike the sigmoid function, it can be used as an activation layers in the hidden layers of CNN structures. At the same time, the Hard sigmoid function is used in loss function calculations and weight updates during the training process of the model. Figure 2.4 shows the hard sigmoid function and defined mathematically in Equation (2.1).

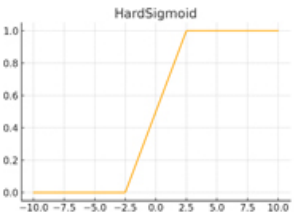


Figure 2.4: Graph of Hard Sigmoid.

$$f(x) = \begin{cases} 0 & \text{if } x \leq -2.5 \\ 1 & \text{if } x \geq 2.5 \\ \frac{x}{5} + 0.5 & \text{if } -2.5 < x < 2.5 \end{cases} \tag{2.1}$$

- **Hard Swish:**

  Hard Swish is a piecewise linear activation function that combines the properties of ReLU and linear functions. It is computationally efficient and has a smoother gradient than ReLU, making it suitable for deep neural networks, as depicted in Figure 2.5 and defined mathematically in Equation (2.2).

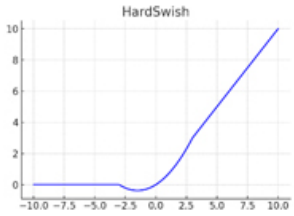


Figure 2.5: Graph of Hard Swish.

$$f(x) = \begin{cases} 0 & \text{if } x \leq -3 \\ x & \text{if } -3 < x < 3 \\ 6x - 9 & \text{if } x \geq 3 \end{cases} \tag{2.2}$$

2.1.1.2.4 Pooling Layers

Pooling layers generally downsample the feature maps after the CNN layers without losing any features. At the same time, since they shrink the feature maps, they prevent overfitting and effectively reduce computational costs. Commonly, two pooling layers are used: average pooling and max pooling layers.

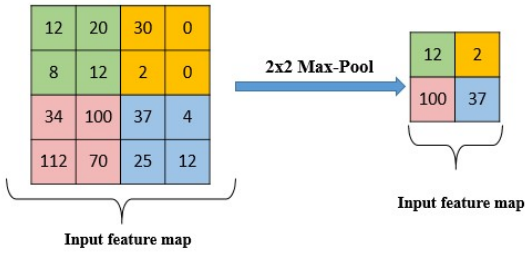


Figure 2.6: Illustration of max pooling.

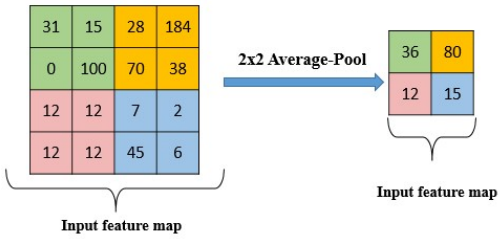


Figure 2.7: Illustration of average pooling.

- **Max Pooling:**
  Max pooling reduces the size of the feature map by taking the maximum value in the selected window size, as shown in Figure 2.3 and Equation 2.3. Still, it helps preserve essential features in the feature map during this size reduction.

$$Max\ Pooling(x, y) = \max_{i,j}(I(x + i, y + j)) \tag{2.3}$$

- **Average Pooling:**
  On the other hand, like max-pooling average pooling also reduces the size of the feature map, but this time, by taking the average value in the selected window size, as shown in Figure 2.7, but it also helps preserve essential features in the feature map during this size reduction. It can be mathematically represented as:

$$Average\ Pooling(x, y) = \frac{1}{N}\sum_{i,j}(I(x + i, y + j)) \tag{2.4}$$

2.1.1.2.5 Fully Connected Layers

Fully connected layers are generally used at the end of the neural network because these layers are used for classification or objection to finding the pixel coordinates of the bounding boxes. In Figure, a fully connected layer network example is given. If W represents the weights of the FC layer, x the input, and b the bias, the output o is given by[2]:

$$o = Wx + b \tag{2.5}$$

These layers are instrumental in classification tasks, where the final FC layer's output size corresponds to the number of classes.

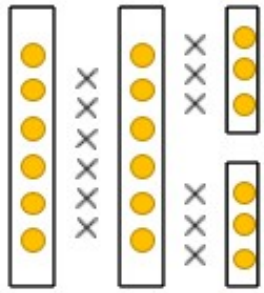


Figure 2.8: Illustration of FC.

2.1.1.2.7 Variants and Architectures:

There are various CNN architectures and variants designed for specific tasks. Notable examples include LeNet[3], AlexNet [4], VGGNet [5], ResNet [6], MobileNetV3[7] and Faster R-CNN [8], which differ in terms of layer depth, filter sizes, and architectural choices, all with mathematical implications for performance

2.1.2 Model Architecture and Design

The model architecture leverages the Faster R-CNN framework combined with both MobileNetV3 and ResNet101 backbones. Faster R-CNN is a leading object detection model that introduces Region Proposal Networks for generating high-quality region proposals, which are then used by the network to detect objects.

MobileNetV3 is a lightweight and efficient neural network structure optimized for mobile and edge devices. It uses depthwise separable convolutions to reduce computational complexity and latency. ResNet101, on the other hand, is a deeper network known for its high accuracy due to its residual learning framework, which helps in training very deep networks.

By incorporating both MobileNetV3 and ResNet101, the model aims to balance high detection performance with computational efficiency, making it ideal for tree detection in aerial imagery. This hybrid approach ensures that the model can handle various challenges posed by different tree types and densities while maintaining the practicality of deployment on devices with limited resources.

2.1.3 Performance Metrics

Performance metrics are important for considering the effectiveness of object detection models like the tree detection model using Faster R-CNN with MobileNetV3. Below are the calculations and significance of each metric:

- **Training and Validation Loss:**
  The specific formulas for training and validation loss can vary depending on the loss functions used. However, a general approach in object detection models like Faster R-CNN combines a classification loss (like Cross-Entropy Loss) with a localization loss (like Smooth L1 Loss).

  1. **Cross-Entropy Loss for Classification:**

$$L_{cls} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{2.6}$$

     Here, M is the number of classes, y is a binary indicator (0 or 1) if class c is the correct classification for observation o, and p is the predicted probability that observation o is of class c.

2. **Smooth L1 Loss for Bounding Box Regression:**

$$L_{box} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i - \hat{t}_i) \tag{2.7}$$

Where t_i and $\hat{t}_i$ are the true and predicted box coordinates, respectively. The Smooth L1 Loss is a less sensitive to outliers version of the L1 Loss.

The overall loss is a weighted sum of these two losses:

$$\text{Loss} = \lambda_{cls} L_{cls} + \lambda_{box} L_{box} \tag{2.8}$$

Where λ are the weights balancing the two losses.

- **Intersection over Union:** IoU is a metric for quantifying the per cent overlap between the target and predicted bounding boxes. The IoU score is computed by dividing the overlap area between predicted and ground truth-bounding boxes by the union area of both boxes. The formula for IoU is:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{2.9}$$

- **Precision and Recall:** Precision is a measure of the accuracy of a predictive model. It refers to the ratio of correctly predicted positive outcomes to the total number of predicted positive outcomes. Recall or sensitivity is the ratio of true positive observations to all positive observations. It is a measure of completeness or quantity. Their formulas are:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.11}$$

where TP is True Positives, FP is False Positives, and FN is False Negatives.

- **Average Precision:**
Average Precision is a metric that condenses the precision-recall curve into one value, representing the average of precision scores at various recall levels. The calculation of AP considers the precision at each threshold, weighting it by the increment in recall relative to the previous threshold:

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{2.12}$$

Where P_n and R_n are the precision and recall at the n-th threshold, respectively.

- **Mean Average Precision:** mAP is the mean of the AP scores for all classes or across different IoU thresholds. For object detection, mAP is often calculated at different IoU thresholds (like 0.5, 0.75) or averaged over a range of IoU thresholds (0.5 to 0.95 in steps of 0.05).

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.13}$$

  Where N is the number of classes or IoU thresholds, and AP_i is the AP for class i or IoU threshold i.

These formulas offer a quantitative measure of the model's performance, considering aspects like the accuracy of the classification, the precision of the bounding box localization, and the balance between detecting as many relevant objects as possible and minimizing false positives.

2.1.4 Data Handling and Image Processing

This research makes use of four different datasets to develop and evaluate the tree detection model. In each dataset, that are all taken from different sources, there are different challenges and qualities, to make sure that an understandable and stable model training comes forward. The datasets are as follows:

- **Oil Palm Tree Dataset:**

  The oil palm tree dataset, obtained from Roboflow, is an extensive COCO format collection specifically used for detecting oil palm trees in plantation settings. It is comprised of 642 training images, which make up 78% of the dataset, along with 184 validation images and 96 test images, accounting for 18% and 10% of the dataset, respectively. The dataset includes a significant number of labels, reflecting the dense annotations per image and the complex nature of the plantation landscapes captured. All images have been consistently resized to 800×800 pixels as a preprocessing step to ensure standardized input dimensions for effective model training.

- **Turkey Dataset:**

  The Turkey Dataset, meticulously crafted and annotated to capture the varied arboreal landscapes of Turkey, comprises an extensive collection of satellite images from specific regions. This dataset includes a total of 1,468 images with 86,971 annotations in COCO format, split into 1,023 training images 78%, 223 validation images 15%, and

222 test images 15%. Notably, this dataset is a personal endeavor aimed at training detection models to recognize the diverse array of tree species found across Turkey's terrain. The dataset specifically avoids augmentations, offering an unaltered reflection of the natural complexity present in real-world satellite imagery.

All images in the dataset are resized to 800 × 800 pixels to satisfy the input requirements of Convolutional Neural Networks (CNNs). This uniformity is crucial for enhancing the generalization capabilities of the models, ensuring that they can accurately identify and classify tree species across varying environmental conditions and the distinct landscapes captured by the satellites. Locations such as Bursa, Adana, Gaziantep, and Antalya are included, each offering its unique tree coverage and soil hues, ranging from dark brown and light brown to beige and yellowish-green tones, adding to the dataset's richness.

2.1.5 Training and Validation Process

The training and validation process is a critical phase where the machine learning model learns to perform its task, in this case, tree detection from aerial imagery. Several key parameters govern this process:

- **Batch Size:**
  This is the number of samples processed before the model's internal parameters are updated. It is a crucial hyperparameter that affects the model's convergence and memory requirements. A smaller batch size often means more updates per epoch and can lead to a more fine-grained learning process, but it may also increase training time. Conversely, a larger batch size can speed up the training process but might lead to less accurate convergence on the optimum.

- **Epoch:**
  An epoch refers to one complete pass through the entire training dataset. The number of epochs in training determines how many times the learning algorithm will work through the entire training dataset. More epochs can lead to a better-trained model and increase the risk of overfitting if not appropriately managed.

- **Learning Rate:**
  This hyperparameter determines the model updates based on the estimated error. Choosing the correct learning rate is crucial as it makes the difference between quickly reaching a good solution and not reaching a solution at all.

- **Early Stopping (Early Patient):**
  Early stopping prevents overfitting in iterative methods such as gradient descent. This method stops the further iteration of the training process when the model's performance on the validation set does not show any improvement for a specific number of epochs, known as "patience."

- **Learning Rate Scheduling (Reducing):**
  This strategy adjusts the learning rate during the training process by reducing it according to a pre-defined schedule or when the model's performance plateaus. This approach can help converge to a better and more robust solution by taking smaller steps when approaching a minimum in the loss landscape. Careful training and validation loss monitoring are required throughout the training process to ensure the model is learning effectively. The validation loss is significant for evaluating the model's performance on unseen data, thus gauging its generalization ability. The aim is to achieve the lowest possible validation loss without the model's performance degrading due to overfitting.

Chapter 3

RELATED LITERATURE

3.1 Using unmanned aerial systems and deep learning for agriculture mapping in Dubai

The study showcases the efficacy of CNNs in vegetation cover analysis with an 85.4% accuracy and high level F1 scores, for the date palm (96.03%) and ghaf trees (94.54%), by leveraging multispectral drone imagery and NDVI integration. This model, though not specified in detail, represents a specialized application of deep learning algorithms for precision agriculture and sustainable land management. [9]

3.2 Resource optimization and image processing for vegetation management programs in power distribution networks

The study assessing U-Net-6 and ResNet-50 networks reveal that ResNet-50 surpassed U-Net-6 in detecting trees, with an accuracy of 93.34% and an IoU of 65.88%. Although U-Net-6 showed a higher mean BF score for trees, it lagged in accuracy and IoU. For background segmentation, U-Net-6 achieved higher accuracy and mean BF scores than ResNet-50, showcasing its potential in certain segmentation tasks. These findings contribute to optimizing resource allocation in vegetation management, reinforcing the potential of deep learning in enhancing power distribution network efficiency.[10]

3.3 Comparison of SSD and Faster R-CNN Algorithms to Detect the Airports with Data Set Which Obtained from Unmanned Aerial Vehicles and Satellite Images

In the study, the researchers evaluated the Single Shot Multibox Detector (SSD) and Faster R-CNN algorithms for detecting airports from various images. The SSD algorithm achieved a detection accuracy of 76.61%, while the Faster R-CNN algorithm outperformed it with an accuracy of 99.52%. This shows the superiority of Faster R-CNN in identifying critical regions in the images used for this study.[11]

3.4 Tree detection from aerial imagery

The study combined classification in the level of pixels and template matching to identify individual tree crowns, obtaining over 90% accuracy in the level of pixels for tree classification. The methods used, examined on a New York dataset, showcased successive advancements, with the baseline accuracy at 86.6%, Cluster I at 89.1%, and Cluster II at 91.7%, demonstrating the effectiveness of the progressive training approaches for differentiating trees from non-trees in aerial images.[12]

3.5 Deep Learning Methods for Tree Detection and Classification

The Faster R-CNN model, applied to tree detection tasks in varying seasonal conditions, yielded F1 scores of 70.3% in March and 61.7% in an August challenge, demonstrating reliable performance throughout the year. In a parallel study, a multi-head ResNet18-based model named 'Soft Teacher' outperformed others like DeepForest and Faster R-CNN, particularly shining with an F1 score of 63.2% during the challenge day, showcasing its robustness against seasonal changes in tree detection from aerial images.[13]

3.6 Individual Tree Detection Based on High-Resolution RGB Images for Urban Forestry Applications

This study explores the efficacy of the Faster R-CNN model along with a Swin Transformer (Swin-T) for urban tree detection, achieving an F1 score of 0.948, outperforming the ResNet50 variant which scored 0.898. The Swin-T model's higher precision and recall suggest it's more suited for the nuanced task of identifying individual trees in complex urban landscapes, offering an advanced tool for urban forestry management.[14]

Chapter 4

SYSTEM DESIGN

4.1 Realistic constraints and conditions

- **Data Availability and Label Quality:**
  Ensuring access to high-quality and accurately labelled training data for optimal model performance.
- **Computational Resources:**
  We require powerful GPUs and substantial data volume for training and implementing the model.
- **Scale and Zoom Level:**
  It necessitates detailed images to enable precise tree detection by the model.
- **Performance Metrics:**
  The importance of selecting suitable metrics to accurately measure the model's effectiveness.
- **Hyperparameter Optimization:**
  Fine-tuning model parameters for optimal performance.

4.2 Cost of the Design

In this project, a computer was the primary tool, with Google Colab Pro utilized for its powerful GPU capabilities. The only expense incurred so far is the cost of Google Colab Pro. In the Table 4.1 cost of design is summarized.

Components	Cost
Google Colab Pro	480 TL
Roboflow	0 TL
Faster R-CNN model	0 TL
Datasets	0 TL

Table 4.1: Cost of the design

4.3 Engineering Standards

- **ISO 16001:2017:**
  Outlines requirements and methods for evaluating object detection systems, ensuring accurate and reliable performance.

- **ISO/IEC 27001:**
  Focuses on a holistic approach to information security, encompassing people, processes, and technology, crucial for managing data in the project.

- **ISO/IEC 15288:2008:**
  Provides guidelines for managing the life cycle of software and systems applicable to software development in this project.

- **ISO 19115:**
  Related to metadata standards for geospatial information, which could be relevant for aerial imagery analysis.

4.4 Details of the System Design

Here, it is provided a comprehensive overview of the system architecture for our project, which focuses on tree detection using object detection systems. The design considers various components: data preprocessing, model architecture, training, and evaluation.

4.4.1 Data Preprocessing

- **Data Format:**
  Our dataset is in COCO format and widely used for object detection tasks. It includes image files and corresponding annotations defining object categories and bounding boxes.

- **Data Transformation:**
  To enhance model robustness, we apply data preprocessing techniques such as resizing images to a standard size and converting them to tensors. These transformations improve the training process.

4.4.2 Model Architecture

- **Faster R-CNN with MobileNetV3 and ResNet101 Backbones:**
  We utilize the Faster R-CNN architecture with a combination of MobileNetV3 and ResNet101 backbones. This approach optimizes the balance between accuracy and computational efficiency for real-time tree detection.

- **Customized Classifier Head:**
  The standard Faster R-CNN classifier head is replaced with a custom head that predicts tree categories specific to our task. This head adapts the model for binary tree detection.

4.4.3 Training

- **Training Data:**
  Every dataset is split into two sections - training and validation subsets using different ratios. Each set contains a representative sample of tree images.

- **Batch Size:**
  We adjusted the batch size to 4 based on GPU memory constraints.

- **Training Duration:**
  The model is trained for 100 epochs to ensure convergence and fine-tuning. The model was optimized using SGD optimizer with a learning rate of 0.005 and weight reduction. The loss function incorporates classification and bounding box regression losses.

4.4.4 Evaluation

- **Validation:**
  It is important to prevent overfitting and that is why we evaluate the model on the validation images of dataset.

- **Mean Average Precision:**
  We assess the model's accuracy using mAP scores at different Intersections over Union (IoU) thresholds (0.25, 0.50, and 0.75). Additionally, we calculate the mAP across the range of IoU between 0.50 and 0.95.

4.4.5 Model Management

- **Model Checkpoints**
  We save model checkpoints after every epochs to ensure reproducibility and reusability.

- **Early Stopping:**
  If performance plateaus after 10 epochs of training, we employ early quitting with a patience value.

- **Learning Rate Scheduling:**
  To fine-tune training, the learning rate is lowered by a factor of 0.1 with a minimum threshold of 5e-7. [15]

4.4.6 Results Tracking

- **TensorBoard:**
  We utilize TensorBoard to visualize training and validation losses, mAP scores, and learning rate changes.

- **Model Selection:**
  The top-performing models based on mAP@50-95 scores are tracked and retained. Our system design ensures that our tree detection model is robust, efficient, and capable of delivering accurate results for tree detection in various scenarios. The combination of data preprocessing, model architecture, training strategies, and evaluation methods contributes to the success of our project.

Chapter 5

METHODS

5.1 Dataset

This section shows the details of the datasets used in this study, information about the test images, validation, and training, number of total labels, and the sources of the each dataset.

Dataset	Training	Validation	Test	Total Labels
Turkey	1023	223	222	86971
Oil Palm Tree	642	184	96	9874

Table 5.1: Summary of Datasets

5.1.1 Data Loader

The DataLoader function is a pivotal element in the training pipeline, handling the input of data to the model. It offers efficient data management capabilities by batching, shuffling, and loading the data, which is crucial for the model to generalize well. Specifically, the DataLoader is configured to shuffle the training dataset at each epoch to prevent the model from learning any ordering, a process that contributes to better model performance.

In Appendix A.1, we detail the implementation of the DataLoader function. The code segment demonstrates the instantiation of the DataLoader with a custom collate function, which is necessary for batching variable-sized data or applying specific transformations. The ‘train_loader‘ and ‘valid_loader‘ are created with specified batch sizes and shuffling options. The ‘train_loader‘ shuffles the data to ensure randomization, while the ‘valid_loader‘ does not, as shuffling is generally not required for validation or testing.

5.1.2 Sample Images From Datasets

- **Oil Palm Tree Dataset:**

Figure 5.1: Samples of the oil palm tree dataset

- **Turkey Dataset:**

Figure 5.2: Samples of the Turkey dataset

5.2 Model Architecture

The architecture employed in this study is a synergy of MobileNetV3, ResNet101, and Faster R-CNN, implemented using the Keras library. MobileNetV3 and ResNet101 serve as complementary backbones for feature extraction, combining their unique strengths to enhance the model's performance.

MobileNetV3 is utilized for its lightweight depthwise separable convolutions, which significantly reduce the model size and computational cost without compromising on performance. It incorporates novel architecture design techniques, such as Squeeze-and-Excitation blocks and Hard Swish activation functions, optimized for mobile and edge devices.

ResNet101 provides robust feature extraction capabilities with its deep residual network structure. ResNet101's architecture includes bottleneck blocks with identity shortcuts, batch normalization layers, and ReLU activation functions, making it highly effective for capturing intricate features within images, though it does come with a higher computational cost compared to MobileNetV3.

Faster R-CNN is leveraged as the detection framework that utilizes the features provided by both MobileNetV3 and ResNet101 to detect and classify objects within the image. It introduces a Region Proposal Network that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. Faster R-CNN is known for its speed and accuracy in detecting objects, making it an excellent choice for real-time applications.

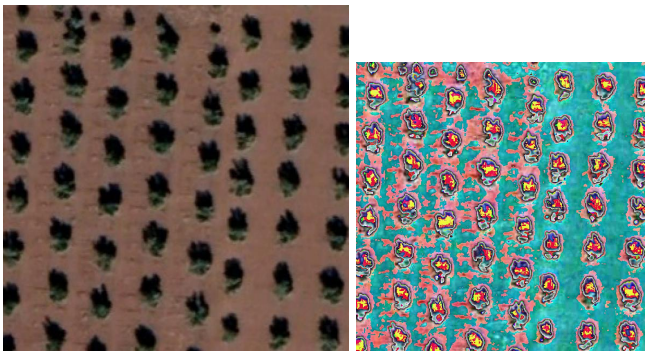
This model is implemented in Keras, a high-level neural networks API written in Python and capable of running on top of TensorFlow. Keras is renowned for its user-friendliness, modularity, and extensibility, which makes the deployment of complex models like MobileNetV3, ResNet101, and Faster R-CNN straightforward for various machine learning tasks. Utilizing Keras also facilitates the model's integration with mobile applications, where TensorFlow Lite can be used for deployment on mobile devices, harnessing the compact and efficient nature of MobileNetV3 alongside the powerful feature extraction of ResNet101.

5.2.1 Layers of the Model

The model is generated from several layers, each with a specific function in the detection process.

5.2.1.1 Transform Layers of the Model

Preprocessing is the first step in our model's pipeline. We first normalized the input images using the mean values [0.485, 0.456, 0.406] and standard deviation values [0.229, 0.224, 0.225]. These are the values commonly used for models that are trained previously on the ImageNet dataset, additionally help in stabilizing the learning process. The GeneralizedRCNNTransform module ensures that the images are resized to fit within the specified minimum and maximum size constraints, which in this case are 800 and 1333 pixels, respectively. This resizing is performed using a bilinear interpolation method, which preserves the quality of the image during the scaling process.



(a) (b)

Figure 5.3: Input (a) and output (b) the transform layers of the model

5.2.1.2 Backbone Layers of the Model

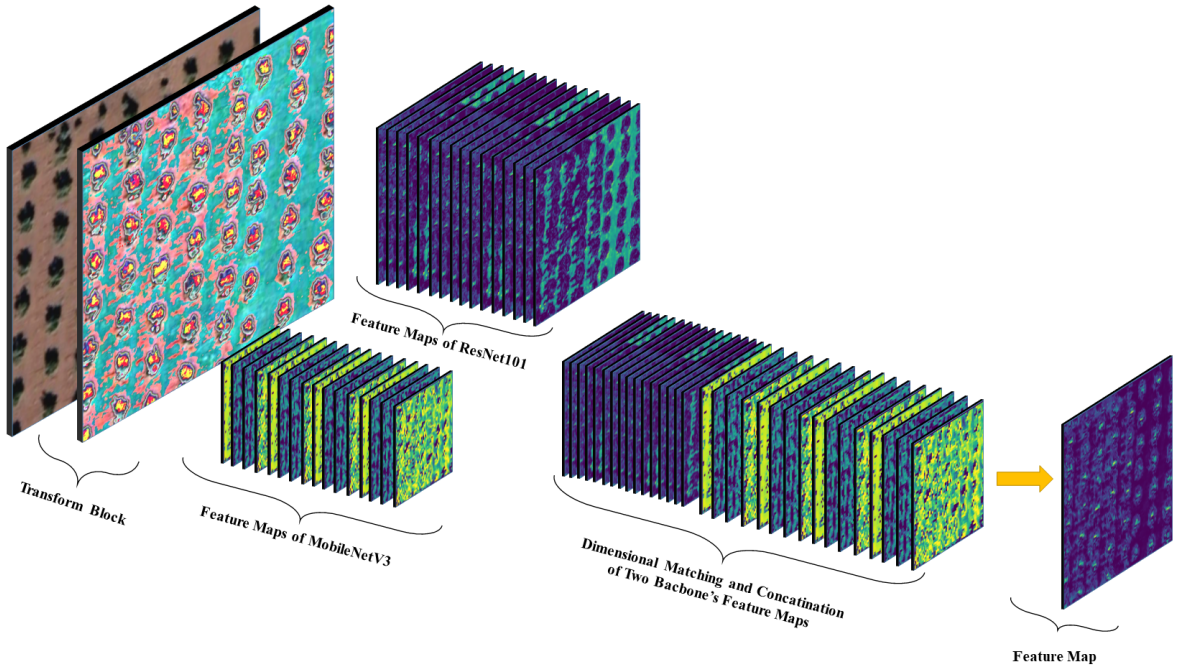


Figure 5.4: Detailed illustration of the model architecture.

The backbone architecture of the model plays a pivotal role in feature extraction. It consists of two parallel backbones: MobileNetV3 and ResNet101. These backbones are combined to leverage the strengths of both architectures for effective feature extraction and processing.

MobileNetV3 Backbone The MobileNetV3 backbone begins with a convolutional layer (Conv2d) with a 3×3 kernel, a stride of 2, and padding of 1. This is followed by batch normalization (BatchNorm2d) and Hardswish activation. The subsequent layers consist of Inverted Residual blocks. Each block includes a depthwise separable convolution and often integrates SqueezeExcitation blocks, which adaptively recalibrate the channel-wise feature responses by modeling the interdependencies between channels. These layers efficiently capture and process spatial hierarchies within the data.

The detailed structure of the MobileNetV3 backbone is as follows:

- Initial Conv2dNormActivation layer.
- A sequence of Inverted Residual blocks with varying configurations, including depthwise convolutions, pointwise convolutions, batch normalization, and ReLU or Hardswish activations.
- SqueezeExcitation blocks nested within some Inverted Residuals to enhance channel interdependencies.
- Final Conv2dNormActivation and AdaptiveAvgPool2d layers.

ResNet101 Backbone The ResNet101 backbone is a deep residual network that includes several residual blocks with skip connections. These connections help mitigate the vanishing gradient problem, ensuring the model can learn effectively even with a large number of layers. ResNet101 captures high-level features and maintains gradient flow throughout the network.

The detailed structure of the ResNet101 backbone includes:

- An initial Conv2d layer with a 7×7 kernel, stride of 2, and padding of 3, followed by batch normalization and ReLU activation.
- A MaxPool2d layer with a kernel size of 3 and stride of 2.
- Four main stages, each consisting of multiple Bottleneck blocks. Each Bottleneck block includes:
  - A sequence of Conv2d layers with batch normalization and ReLU activation.
  - A skip connection that bypasses one or more layers.
  - Downsample layers in certain blocks to reduce the spatial dimensions.
- A final AdaptiveAvgPool2d layer and a fully connected layer for dimensionality reduction.

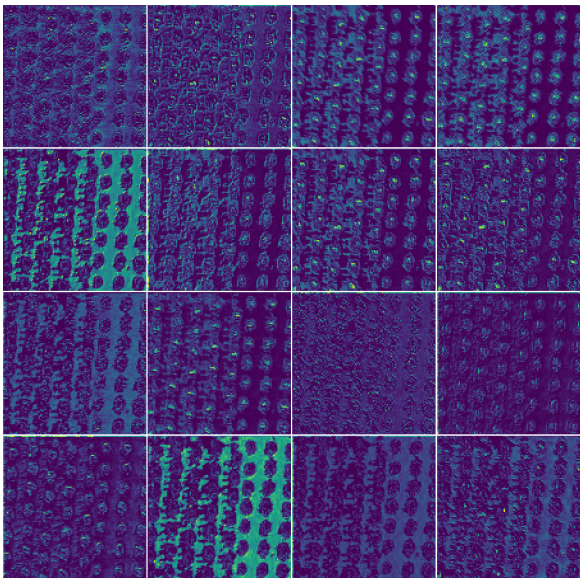


Figure 5.5: Outputs of the convolution layers.

Combined Feature Extraction The feature maps generated by MobileNetV3 and ResNet101 are concatenated to create a comprehensive feature representation. This combination ensures that the lightweight and efficient feature extraction capabilities of MobileNetV3 are complemented by the deep and robust feature representations of ResNet101.

The combined architecture of MobileNetV3 and ResNet101 ensures a balance between efficiency and performance, leveraging the strengths of both networks to transform the input data into a rich feature map, ready for further processing by the subsequent parts of the model architecture. The detailed concatination and combination algorithms are illustrated in below figure.

5.2.1.3 RPN Structures of the Model

The RPN is an innovative layer designed to generate object proposals. It begins using an Anchor Generator, which creates a set of anchor boxes over the feature map that serve as potential object locations. These anchor boxes are diverse in scale and aspect ratio, covering a wide range of object sizes and shapes.

Following the anchor generation, the RPN applies a shared convolutional layer across the feature map to get the objectness score and bounding box adjustments in every anchor. This shared layer ensures that the learned features are translation-invariant, thus enabling the model to detect objects regardless of their position in the image.

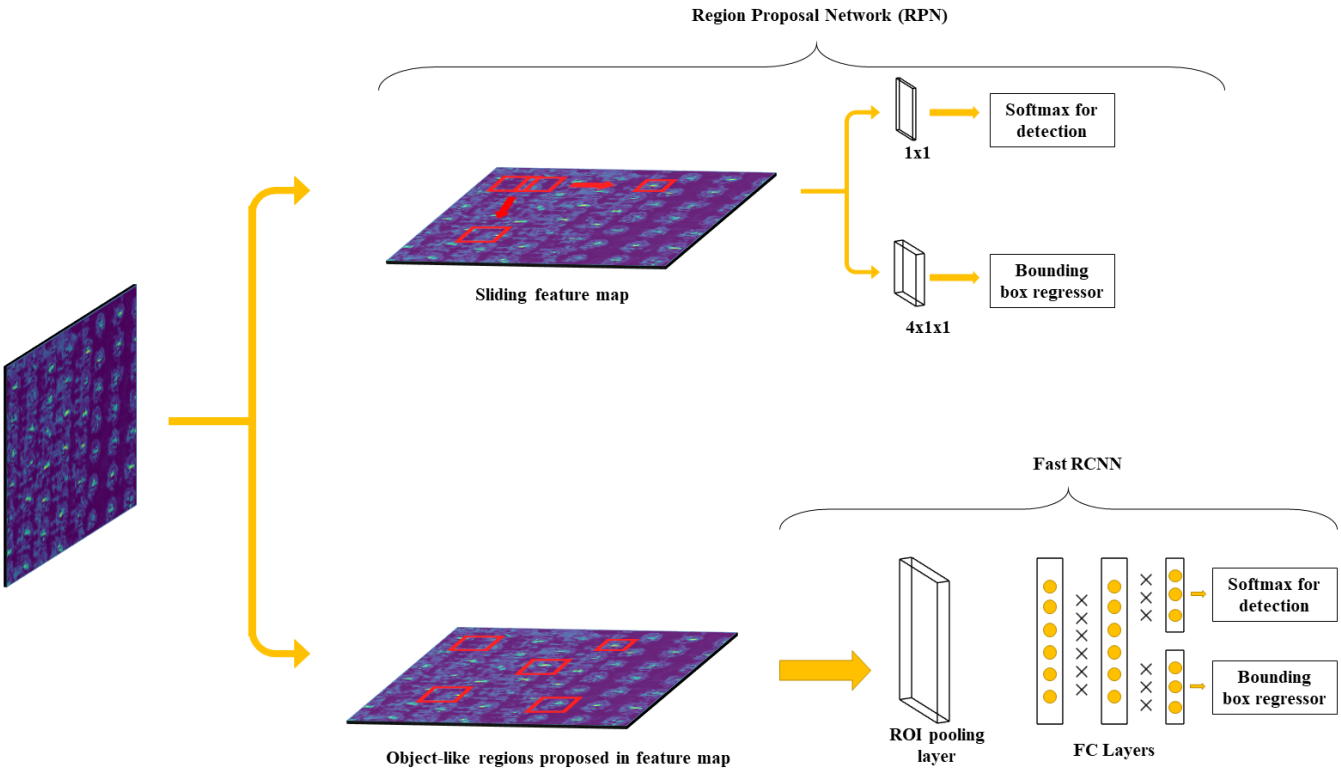


Figure 5.6: Detailed illustration of the RPN architecture.

The classification layer (cls_logits) in the RPN tries to calculate the probability of an anchor being an object or background, while the regression layer (bbox_pred) refines the coordinates of the anchor to better fit the proposed object. The combination of these predictions results in a set of refined bounding boxes that are passed to the RoI pooling layer, moving forward in the detection pipeline.

The RPN is a pivotal part of the model's architecture, bridging the gap between feature extraction and object detection. Thanks to the ability to share features with the detection network, it greatly accelerates perception time and reduces the computational load by limiting the number of proposals.

5.2.1.4 ROI Layers of the Model

The Region of Interest (RoI) layers function as the decision-making component of the model, taking the candidate object locations proposed by the RPN and extracting relevant features for classification and bounding box regression. Thus, achieved through a process

known as RoI pooling, which converts the irregularly shaped regions into a fixed-size feature map suitable for feeding into fully connected layers.

The RoI pooling layer, specifically the MultiScaleRoIAlign layer in our model, performs spatial quantization on the input feature maps and the proposed regions to produce uniformly sized output feature maps. This layer handles proposals of various sizes and scales by aligning and pooling features from different levels of the FPN. As a result, each object, irrespective of its size, is represented by a feature map of the same size, ensuring consistent processing in subsequent layers.

Once the features are pooled, the TwoMLPHead module, consisting of two fully connected layers, takes over. It processes the pooled features to further refine the representations. The first fully connected layer, termed 'fc6', transforms the input feature vector into a higher dimensional space, allowing the model to learn more complicated features. The second layer, 'fc7', continues this transformation and prepares the features for the final classification and regression tasks.

The box_predictor module then receives the output from the TwoMLPHead. It consists of two parts: the cls_score layer, which predicts the probability values of each RoI for the classes, and the bbox_pred layer, which outputs the bounding box regression offsets for each RoI. These predictions are used to classify the objects and refine their locations within the image.

The RoI layers are essential for translating the proposals into precise object detections, making them a critical part of the detection pipeline. The combination of RoI pooling and the subsequent fully connected layers allows the model to make exact predictions on the position and category of each object in the image.

5.2.2 Training and Validation Process

For classification, the model is trained using cross-entropy loss, and for bounding box regression, it uses smooth L1 loss. By employing distinct weights to modify the ratio of these two losses, the model may efficiently acquire knowledge for both tasks at the same time.

Parameter	Value
Batch Size	4
Number of Epochs	60
Learning Rate	0.005
Momentum	0.9
Weight Decay	0.005
Early Stopping Patience	10 epochs
Learning Rate Reduction Patience	3 epochs
Learning Rate Reduction Factor	0.1
Minimum Learning Rate	5×10^{-7}

Table 5.2: Summary of Hyperparameters and Training Criteria

5.3 Training Process

5.3.1 Hyper Parameters

5.3.2 Training Duration

The training duration for the model is primarily determined by the number of epochs, which is set to 60. This duration is subject to modification based on early stopping criteria or performance metrics observed during training. The 'train' function is elaborated in Appendix A.2.

5.3.3 Optimizer

The model uses the SGD [16] optimizer, which is set to 0.005 as the initial learning rate, 0.9 momentum, and weight decay 0.005.

5.3.4 Learning Rate Scheduling

A type of learning rate planning strategy is used when adjusting the pace during training. Through the timer the learning rate is reduced by a factor of 0.1 if there's no improvement for 3 consecutive epochs, with a minimum learning rate set at 5×10^{-7}.

5.3.5 Early Stopping

Early stopping is employed to prevent overfitting. Training is stoped if there's no improvement in validation loss for 10 consecutive epochs.

5.4 Validating Process

5.4.1 Validation Duration

The validation process is an essential part of the training cycle, conducted at the end of each training epoch. This process provides information about the generalization capabilities of the model while evaluating its performance on the validation dataset in the dataset and helps prevent overfitting.

Specifications of the verification procedure are detailed in Appendix A.3, where the 'verification' function is outlined. This function iteratively processes the validation dataset and calculates the validation loss, indicating the model's accuracy on images we have yet to expose.

5.5 Model Management

5.5.1 Saving Best Model

In addition to saving the best model based on validation loss, the training process also entails the preservation of models based on their mean Average Precision (mAP) scores. This approach allows for the retention of top-performing models in terms of their accuracy in object detection tasks.

Models are saved if they are among the top 10 in terms of their mAP scores, with ensuring that only the top models are retained for each category of mAP scores. This segment of the code illustrates the criteria for saving models based on their mAP scores. On the other hand because of the do not missing any model every epochs are saved in different list as seperatelly. This saving best model algotithm is detailed in Appendix A.4.

5.6 Metric Monitoring

5.6.1 TensorBoard

In the training and validation process, TensorBoard is employed to monitor and log key metrics. This tool is instrumental in tracking the model's performance over time, providing visual insights into various metrics. The specific metrics logged include:

- **Training Loss:**
  Represents the error of the model during the step of training.

- **Validation Loss:**
  Represents the error of the model on the dataset we will use to validate, providing insights into its generalization capability.

- **Mean Average Precision (mAP) Scores:**
  These are logged at different thresholds (25%, 50%, 75%, and 50:95), offering a detailed view of the model's accuracy in object detection at various levels of strictness.

Chapter 6

RESULTS AND DISCUSSION

6.1 Simulation Results

6.1.1 Results of Turkey Dataset

This subsection discusses the performance of different models trained using the Turkey dataset. Visualizations of model performance metrics and sample outputs are presented to provide insights into each model's effectiveness.

6.1.1.1 Results of MobileNetv3 Model

The MobileNetv3 model, known for its efficiency in mobile applications, was applied to the Turkey dataset. Below are the performance metrics and sample output visualizations that demonstrate the model's capability in handling this specific dataset.

Figure 6.1: Sample outputs of the MobileNetv3 model on the Turkey dataset

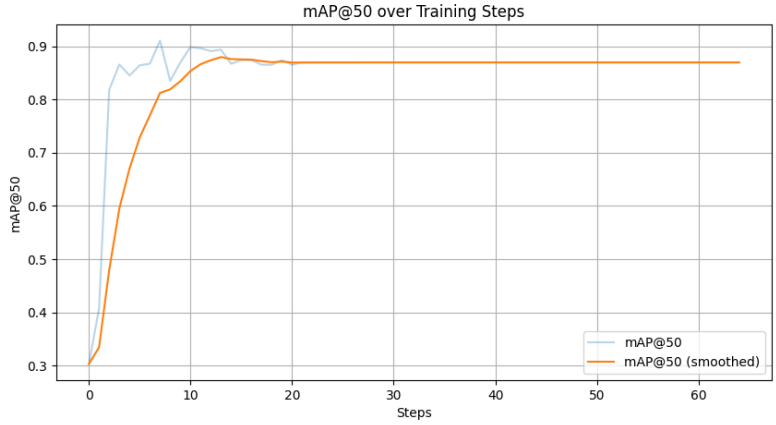


Figure 6.2: Graph of mAP@50 vs epoch

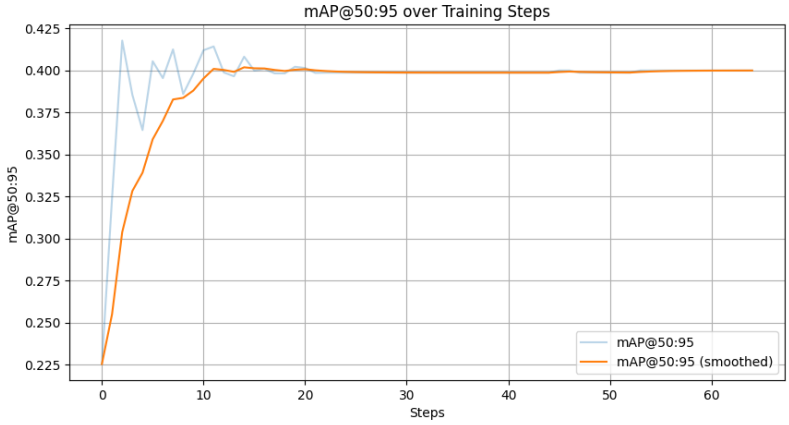


Figure 6.3: Graph of mAP@50:95 vs epoch

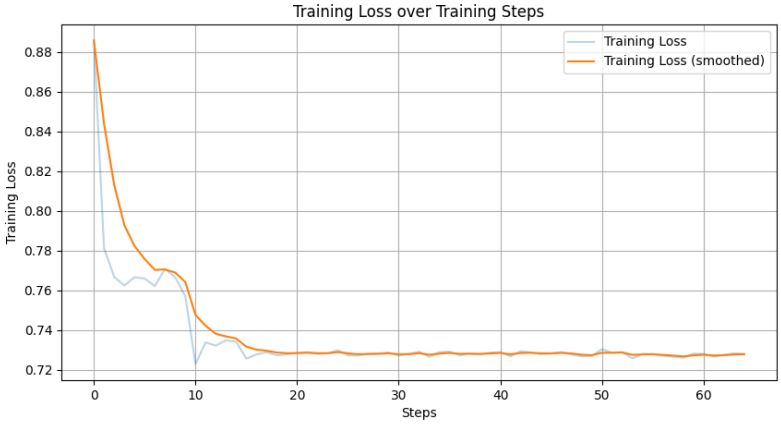


Figure 6.4: Graph of training loss vs epoch

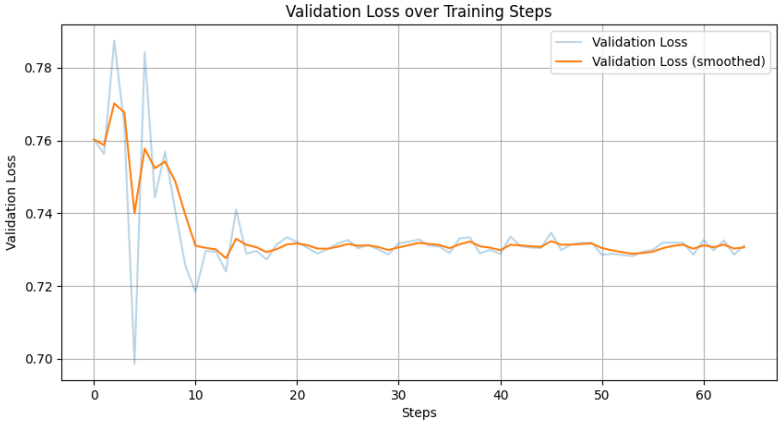


Figure 6.5: Graph of validation loss vs epoch

6.1.1.2 Results of ResNet101 Model

ResNet101, renowned for its deep architecture and accuracy, was utilized for the Turkey dataset. The following figures illustrate the model's performance across various metrics and outputs.

Figure 6.6: Sample outputs of the ResNet101 model on the Turkey dataset

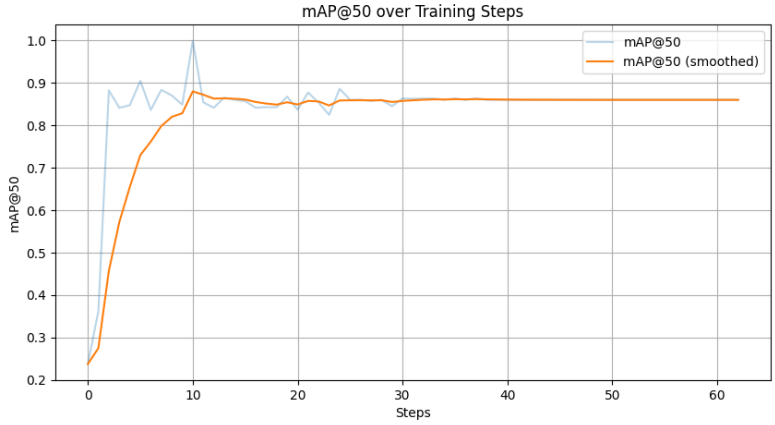


Figure 6.7: Graph of mAP@50 vs epoch

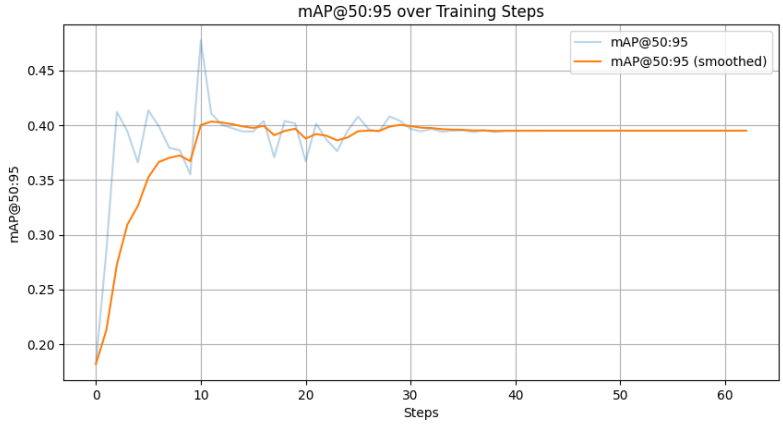


Figure 6.8: Graph of mAP@50:95 vs epoch

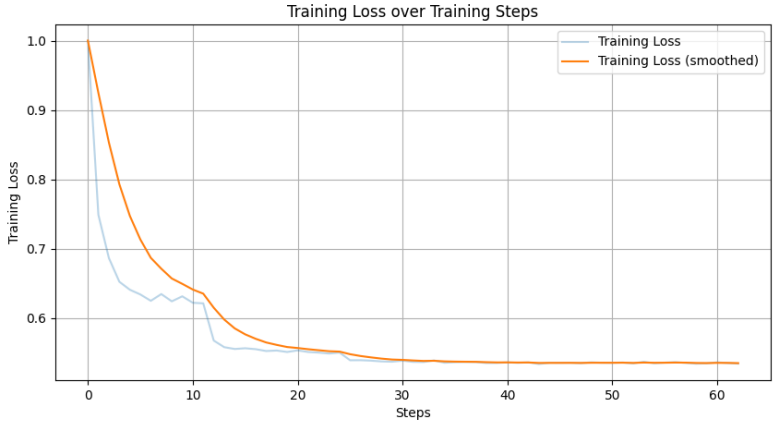


Figure 6.9: Graph of training loss vs epoch

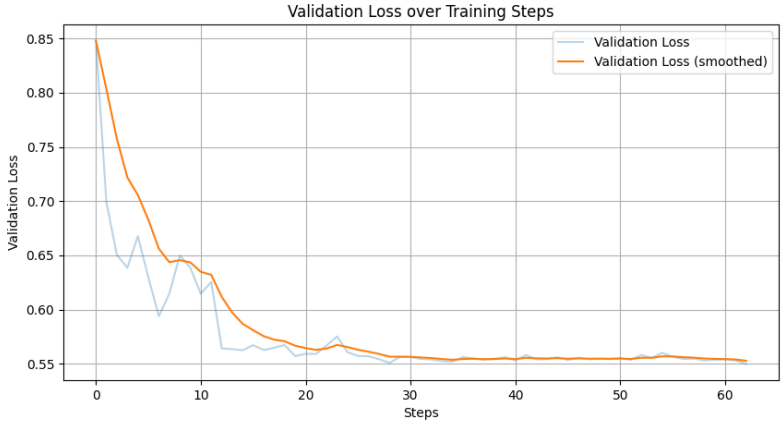


Figure 6.10: Graph of validation loss vs epoch

6.1.1.3 Results of Merged Model MobileNetv3 + ResNet101

The merged model combines the strengths of MobileNetv3 and ResNet101 to leverage their complementary benefits. Here are the performance metrics and visual outputs demonstrating the effectiveness of this hybrid approach on the Turkey dataset.

Figure 6.11: Sample outputs of the merged MobileNetv3 + ResNet101 model on the Turkey dataset

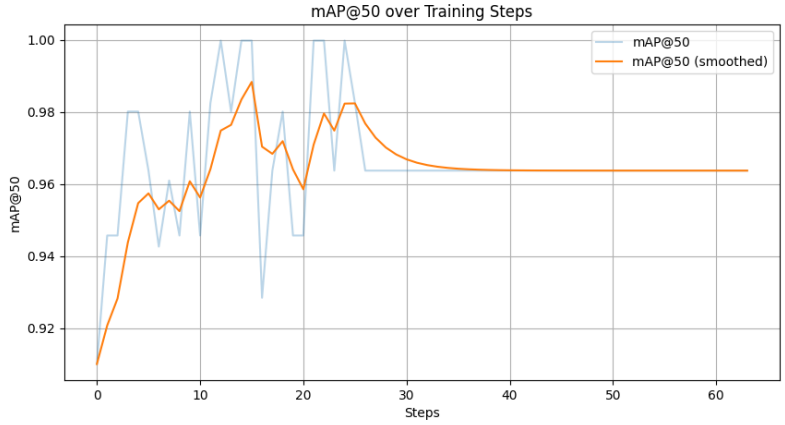


Figure 6.12: Graph of mAP@50 vs epoch

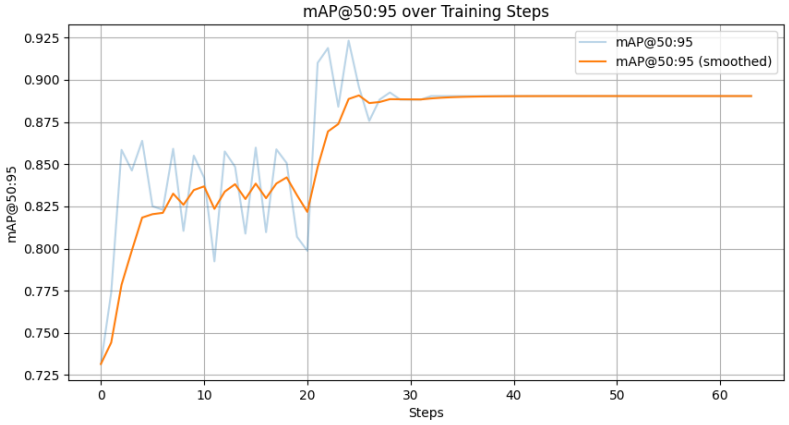


Figure 6.13: Graph of mAP@50:95 vs epoch

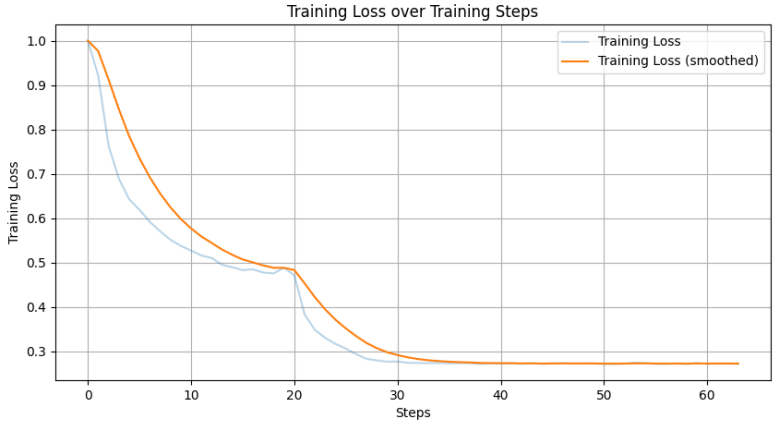


Figure 6.14: Graph of training loss vs epoch

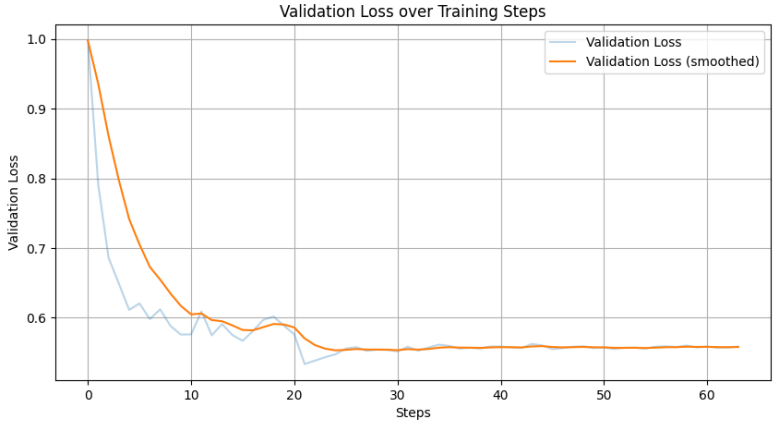


Figure 6.15: Graph of validation loss vs epoch

6.1.2 Results of Oil Palm Tree Dataset

This section illustrates the application of various models to the Oil Palm Tree dataset. Performance metrics and outputs are displayed to highlight each model's ability to process and analyze this dataset effectively.

6.1.2.1 Results of MobileNetv3 Model

MobileNetv3's application to the Oil Palm Tree dataset shows promising results, which are depicted in the following figures. These highlight the model's accuracy and efficiency in handling complex datasets.

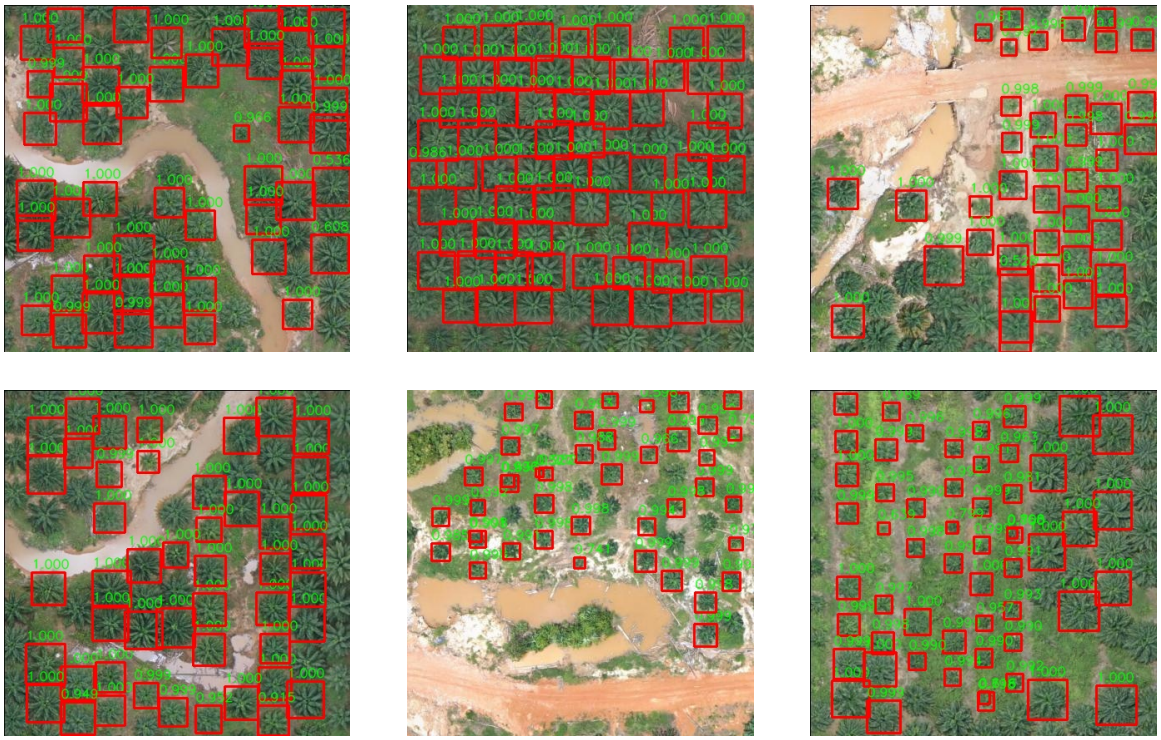


Figure 6.16: Sample outputs of the MobileNetv3 model on the Oil Palm Tree dataset

6.1.2.2 Results of ResNet101 Model

Employing the deep ResNet101 model on the Oil Palm Tree dataset has yielded detailed insights, as shown in the performance metrics and visual outputs below. The depth of this model aids in effectively managing the dataset's complexities.

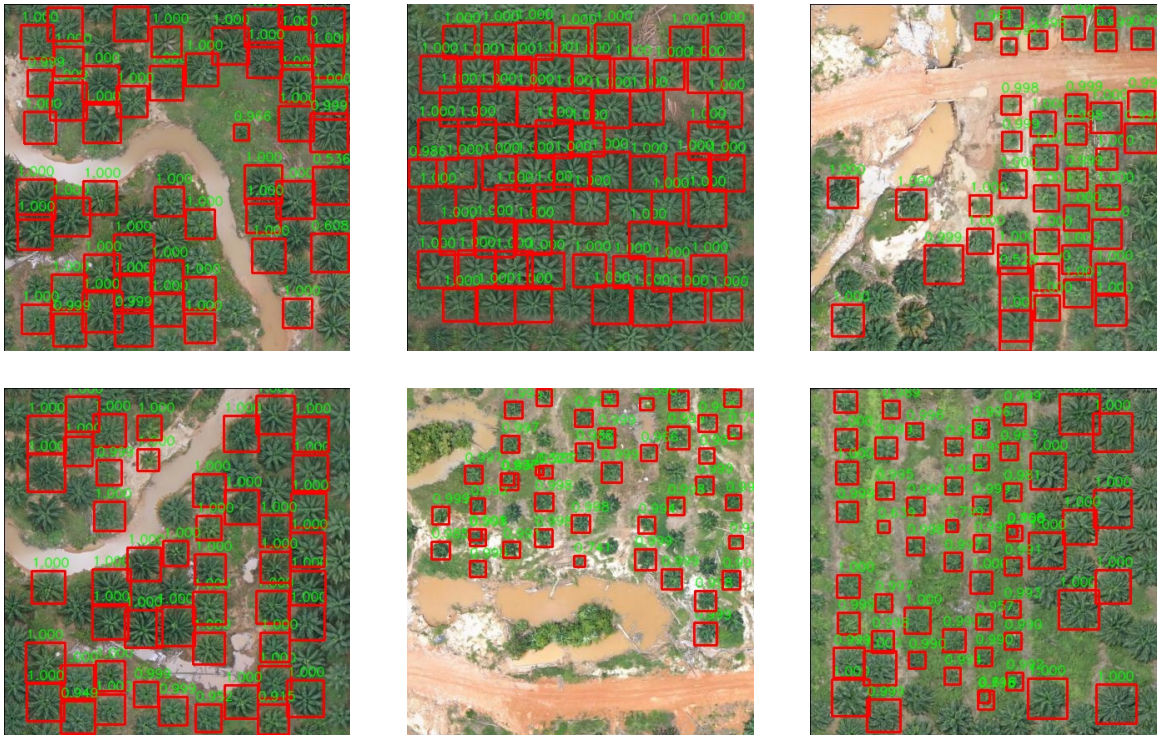


Figure 6.17: Sample outputs of the ResNet101 model on the Oil Palm Tree dataset

6.1.2.3 Results of Merged Model MobileNetv3 + ResNet101

Combining MobileNetv3 and ResNet101's capabilities provides a robust model for the Oil Palm Tree dataset. The merged model's performance metrics and outputs are illustrated below, showcasing enhanced effectiveness.

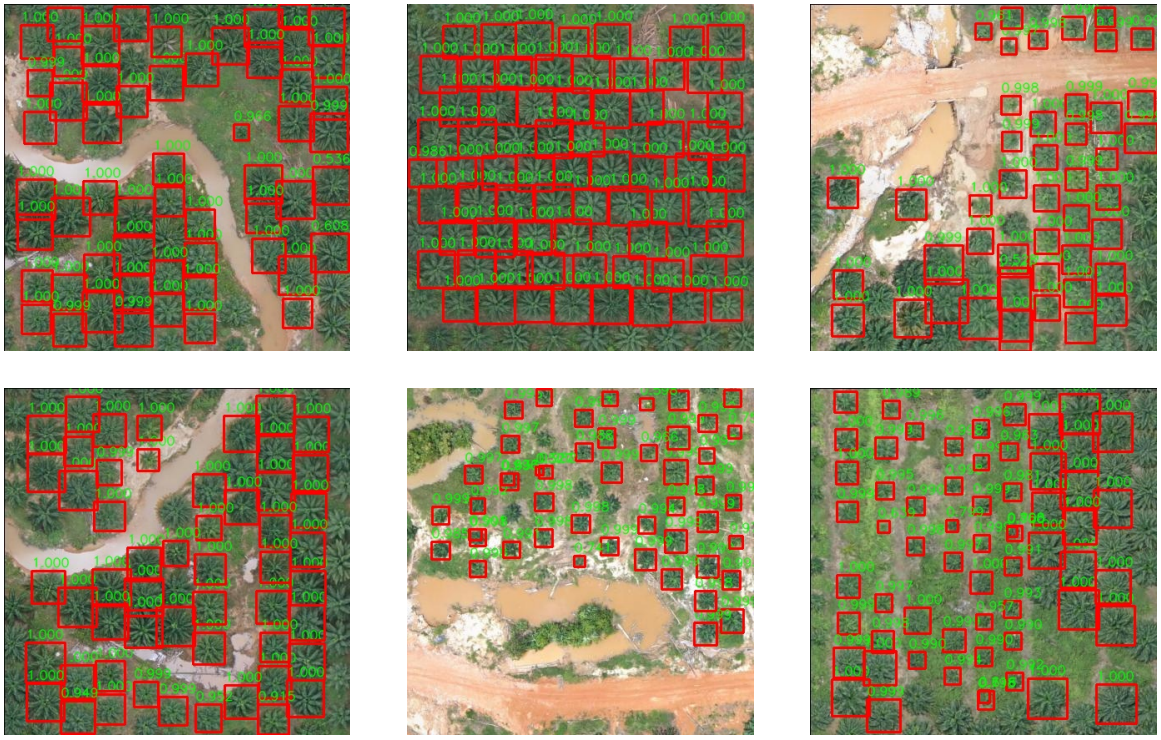


Figure 6.18: Sample outputs of the merged MobileNetv3 + ResNet101 model on the Oil Palm Tree dataset

TURKEY DATASET				
Model Architecture	**Min Train Loss**	**Min Valid Loss**	**mAP@50**	**mAP@50:95**
MobileNetV3+ResNet101	0.5245	0.5348	0.9637	0.8373
ResNet101	0.5344	0.5499	0.8901	0.4950
MobileNetV3	0.7348	0.7369	0.8695	0.3999
OIL PALM TREE DATASET				
Model Architecture	**Min Train Loss**	**Min Valid Loss**	**mAP@50**	**mAP@50:95**
MobileNetV3+ResNet101	0.6257	0.6549	0.9081	0.8741
ResNet101	0.6357	0.6483	0.8901	0.7405
MobileNetV3	0.7549	0.7763	0.8695	0.6504

Table 6.1: Results on Turkey and Oil Palm Tree Datasets

Discussion

Results Analysis of MobileNetv3 and ResNet101 Models on the Turkey and Oil Palm Tree Datasets

The findings from the application of the MobileNetv3 and ResNet101 models, and their merged configuration on the Turkey and Oil Palm Tree datasets, illustrate their strengths and limitations in processing complex remote sensing data for environmental and agricultural applications.

MobileNetv3's Performance: MobileNetv3, tailored for mobile efficiency, showed commendable performance on the Turkey dataset, achieving high mAP@50 scores indicative of precise object localization. However, its efficacy was slightly lower on the more demanding Oil Palm Tree dataset. This could be due to the dataset's intrinsic complexities or the architectural limitations of MobileNetv3 in handling detailed satellite imagery. The training and validation loss graphs demonstrated stable convergence, reflecting the model's robustness against overfitting, which is advantageous in resource-constrained scenarios.

ResNet101's Performance: ResNet101 excelled in capturing intricate details in both datasets, as evidenced by consistently high mAP scores across various thresholds. The depth of ResNet101 enhances its capability for detailed feature extraction, crucial for accurate identification and classification in satellite images. Nonetheless, the training and validation loss graphs indicated some initial convergence challenges, suggesting the need for further hyperparameter tuning or extended training periods to optimize performance.

Merged Model Insights: Combining MobileNetv3 and ResNet101 resulted in a hybrid model that improved robustness and accuracy, particularly noticeable in the Oil Palm Tree dataset. This model effectively merged the efficiency of MobileNetv3 with the depth of

ResNet101, leading to superior performance metrics. Compared to the individual models, the merged model exhibited smoother loss graphs, indicating a more effective learning process, likely due to the synergistic integration of the feature representations from both architectures.

Chapter 7

CONCLUSION

7.1 Conclusion

This research aimed to enhance tree detection in aerial images by utilizing a dual-backbone architecture combining MobileNetV3 and ResNet101 within the Faster R-CNN framework. The comprehensive evaluation across multiple datasets, including the Oil Palm Tree Detection Dataset and Turkey Dataset, demonstrated significant improvements in detection accuracy and computational efficiency.

The dual-backbone approach was meticulously trained and validated, employing techniques such as data augmentation and early stopping to prevent overfitting. The results indicated that the combined architecture achieved impressive mean Average Precision (mAP) scores across various thresholds. For instance, the Oil Palm Tree Detection Dataset achieved a maximum mAP@50 of 0.9081 and mAP@50:95 of 0.8741, while the Turkey Dataset achieved a maximum mAP@50 of 0.9637 and mAP@50:95 of 0.8373. These results showcase the model's robustness and precision in diverse environments.

Additionally, the adaptability of the model was highlighted by its performance on the Turkey and Oil Palm Tree datasets. Despite varying environmental conditions and image complexities, the dual-backbone model consistently demonstrated high detection accuracy, with decreasing training and validation losses indicating effective learning and generalization. This adaptability underscores the model's potential for practical applications in environmental monitoring and sustainable forest management.

In conclusion, the integration of MobileNetV3 and ResNet101 backbones within the Faster R-CNN framework has significantly advanced the field of aerial tree detection. This study not only contributes to academic knowledge but also provides a practical tool for real-world applications in ecological balance and forest conservation projects. The

research findings emphasize the importance of combining deep learning techniques with efficient model architectures to address environmental challenges and promote sustainable management practices.

7.2 Future Work

While the current dual-backbone model exhibits high accuracy and robustness, there is potential for further enhancements. Future work could explore the integration of additional layers in the MobileNetV3 and ResNet101 backbones to increase the model's robustness without significantly impacting computational efficiency. Furthermore, experimenting with advanced data augmentation techniques and incorporating climatic factors could improve the model's adaptability to seasonal changes and different weather conditions.

Another promising direction is the application of the model to real-time monitoring systems. Optimizing the model for deployment on mobile and edge devices using TensorFlow Lite can enhance its practicality for on-the-go environmental monitoring. Additionally, expanding the dataset to include more diverse geographical regions and tree species could further validate and refine the model's accuracy and generalizability.

Overall, these improvements could solidify the model's position as a valuable tool for environmental monitoring, contributing to more effective and sustainable forest management practices.

Appendix A

A.1

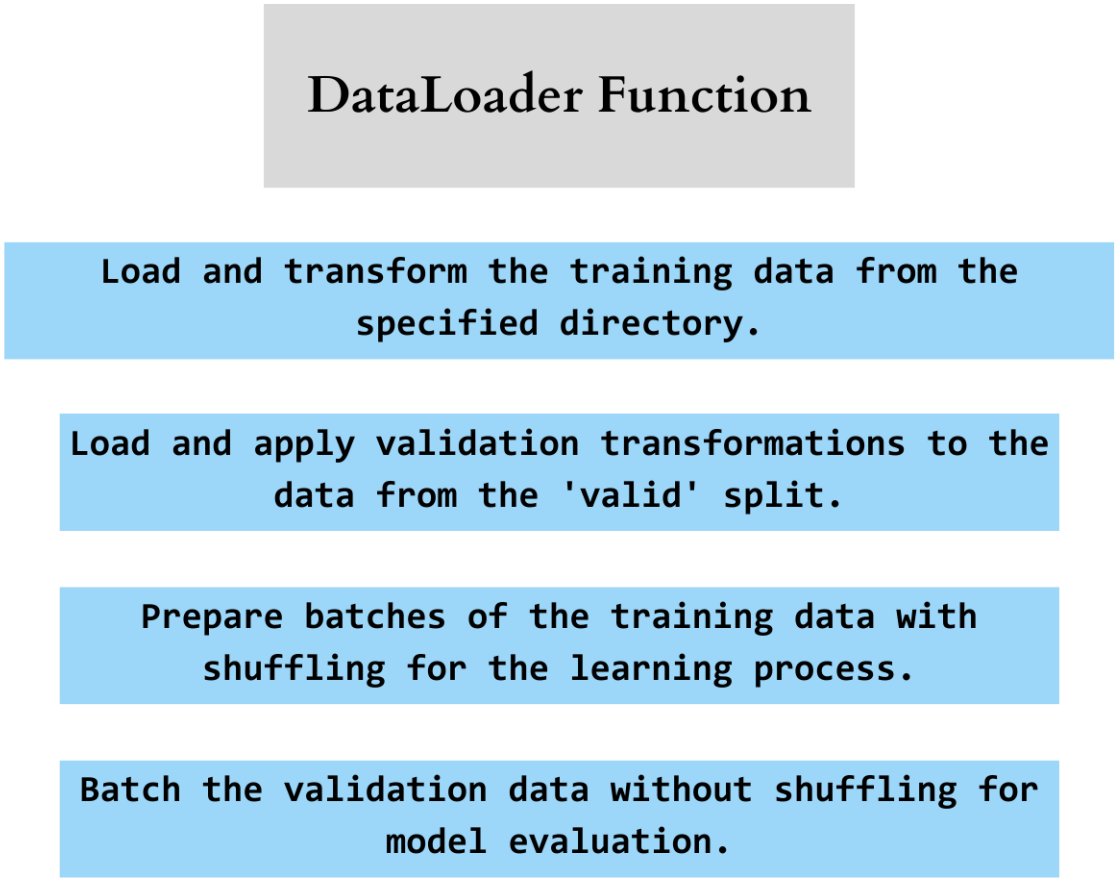


Figure A.1: Flow diagram of DataLoader function

A.2

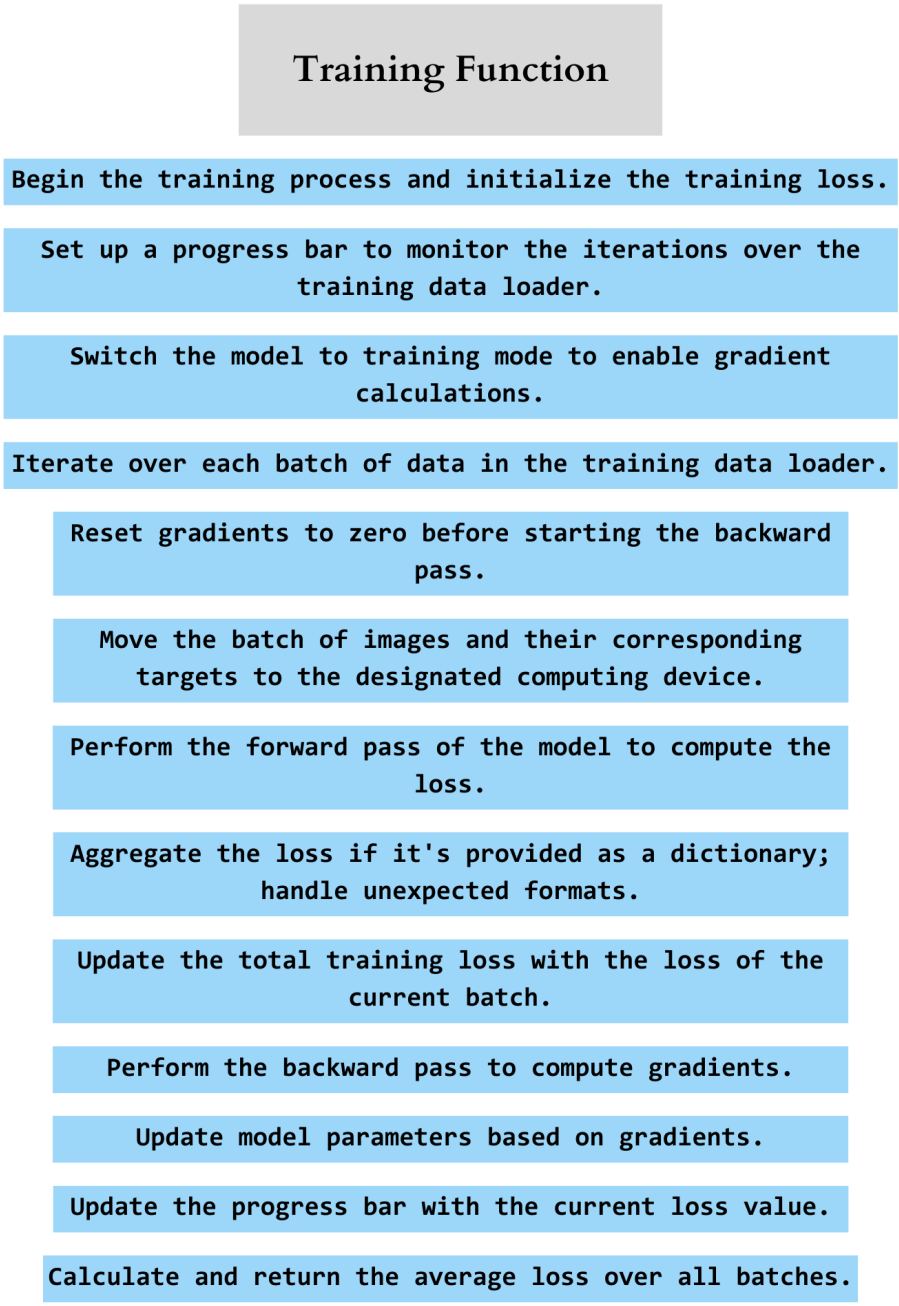


Figure A.2: Flow diagram of Training function

A.3

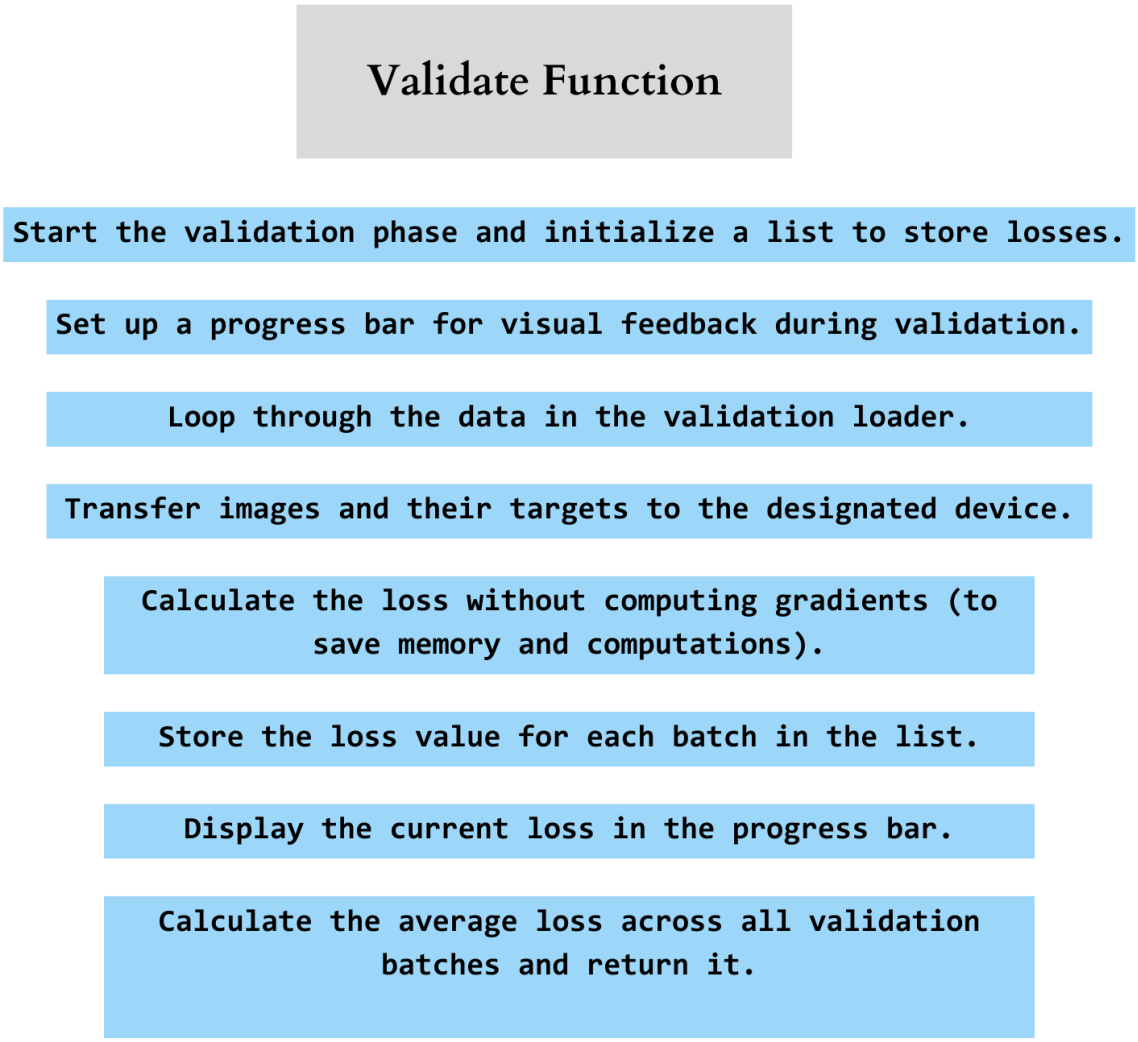


Figure A.3: Flow diagram of Validate function

A.4

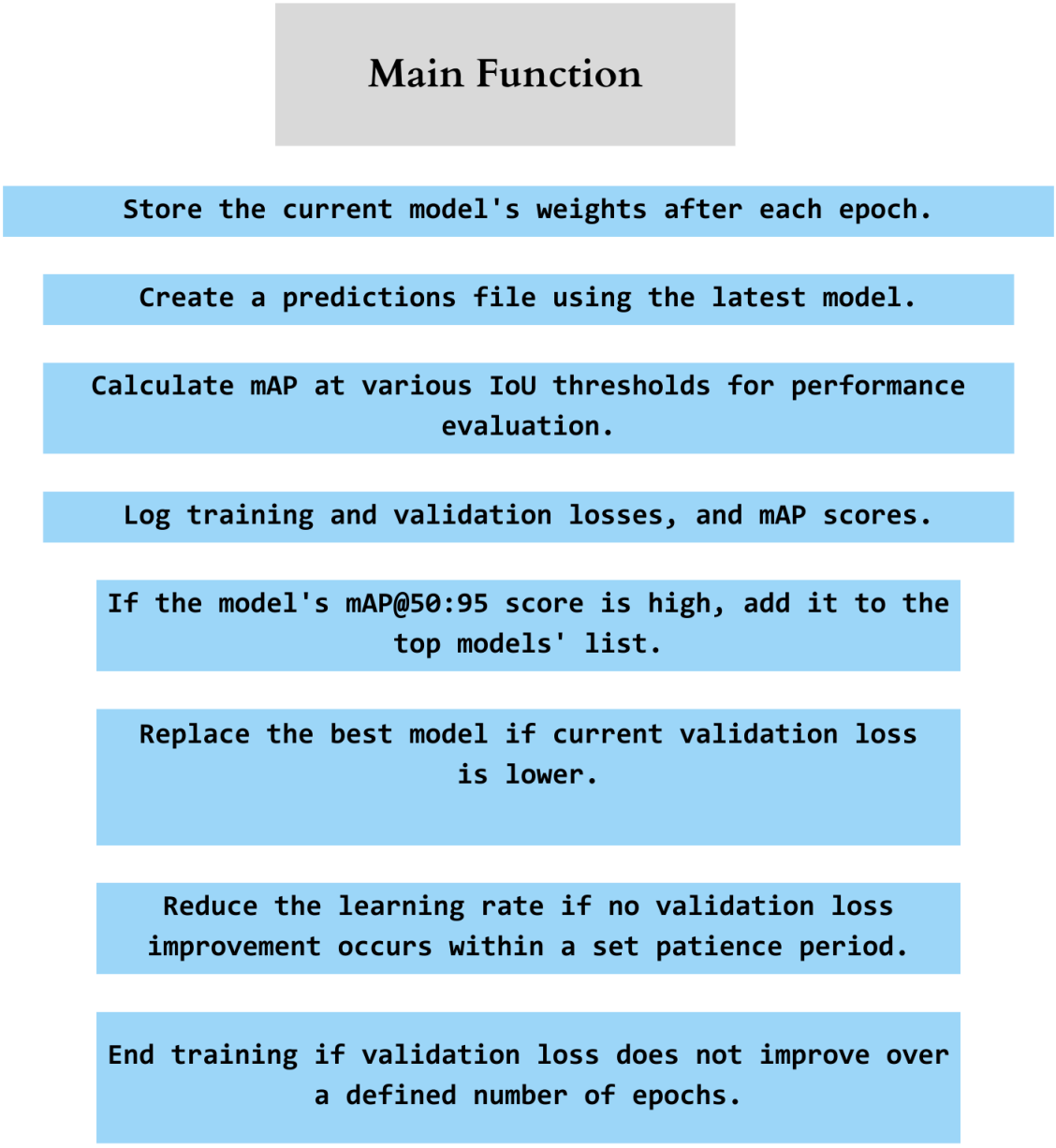


Figure A.4: Flow diagram of Main function

REFERENCES

[1] Bulgaria: The ministry of agriculture and food announces a competition for the logo and slogan of the national program for prevention and reduction of food loss. *MENA Report*, 2021.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, 2012.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[9] Lala El Hoummaidi, Abdelkader Larabi, and Khan Alam. Using unmanned aerial systems and deep learning for agriculture mapping in dubai. *Heliyon*, 7(e08154), 2021.

[10] Olga Jumbo and Ramin Moghaddass. Resource optimization and image processing for vegetation management programs in power distribution networks. *Applied Energy*, 319:119234, 2022.

[11] Muhammed Taha Zeren, Sabahattin Kerem Aytulun, and Yasin Kırelli. Comparison of ssd and faster r-cnn algorithms to detect the airports with data set which obtained from unmanned aerial vehicles and satellite images. *European Journal of Science and Technology*, (19):643–658, 2020.

[12] Lin Yang, Xiaqing Wu, Emil Praun, and Xiaoxu Ma. Tree detection from aerial imagery. 2009. This work was done at Google Inc.

[13] Yang Zhang, Yizhen Wang, Zhicheng Tang, Zhenduo Zhai, Yi Shang, and Reid Viegut. Deep learning methods for tree detection and classification. In *2022 IEEE 4th International Conference on Cognitive Machine Intelligence (CogMI)*. IEEE, 2022.

[14] Lishuo Zhang, Hong Lin, and Feng Wang. Individual tree detection based on high-resolution rgb images for urban forestry applications. *IEEE Access*, 10:46589–46597, 2022.

[15] Adam Pluta et al. Detecting pipeline pathways in landsat 5 satellite images with deep learning. *Energies*, 14(18):5642, 2021.

[16] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.