UNIVERSITY OF PAVIA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL, COMPUTER
AND BIOMEDICAL ENGINEERING

MASTER'S DEGREE IN COMPUTER ENGINEERING

MASTER THESIS

Object detection in ultra-high resolution satellite SAR data by adaptation of a YOLO model

Rilevamento di oggetti nei dati SAR satellitari ad altissima risoluzione mediante l'adattamento di un modello YOLO

Candidate: Cristian Andreoli

Supervisor: Prof. Fabio Dell'Acqua

Academic Year 2023/2024

Abstract

This thesis focuses on the application of object detection techniques to high-resolution SAR (Synthetic Aperture Radar) images acquired from ICEYE satellites, highlighting the peculiarities and intrinsic challenges of these data—such as acquisition geometry, double bouncing phenomena, and the presence of speckle—that make the detection of small targets particularly complex.

In the first phase, the work analyzes the use of the YOLO family of algorithms, employed in various configurations (GRD and CSI images) and at different resolutions, to evaluate performance in terms of speed and accuracy in object detection. Subsequently, an approach based on a Transformer foundation model, named Clay, is examined. This model is adapted to SAR images through a fine-tuning process supported by a custom loss function that integrates Complete Intersection over Union (CIoU) and Mean Squared Error (MSE) for bounding box regression, alongside components of Binary Cross Entropy (BCE) for classification. In parallel, a ResNet50-based backbone—trained using learning rate scheduling techniques—is also compared to explore the implications of extended training.

The experimental results indicate that while the YOLO approach proves robust in managing the complexities of SAR data, methodologies based on Transformer models require further investigation to fully leverage the unique characteristics of radar imagery. In conclusion, the thesis underscores the complementarity between classical and innovative approaches, outlining future perspectives for optimizing object detection systems in the SAR domain, with potential applications in Earth observation and the monitoring of critical infrastructures and activities.

Sommario

La presente tesi si concentra sull'applicazione di tecniche di object detection a immagini SAR (Synthetic Aperture Radar) ad alta risoluzione, acquisite dai satelliti ICEYE, mettendo in luce le peculiarità e le sfide intrinseche legate a questi dati – quali la geometria di acquisizione, il fenomeno del double bouncing e la presenza di speckle – che rendono particolarmente complesso il rilevamento di piccoli bersagli.

In una prima fase, l'elaborato analizza l'utilizzo della famiglia di algoritmi YOLO, impiegata in diverse configurazioni (immagini GRD e CSI) e a differenti risoluzioni, per valutare le performance in termini di rapidità e accuratezza nell'individuazione degli oggetti. Successivamente, viene esaminato un approccio basato su un foundation model Transformer, denominato Clay, che viene adattato alle immagini SAR tramite un processo di fine-tuning supportato dall'impiego di una funzione di loss custom. Tale funzione integra il Complete Intersection over Union (CIoU) e il Mean Squared Error (MSE) per la regressione delle bounding box, insieme a componenti di Binary Cross Entropy (BCE) per la classificazione. Parallelamente, si confronta anche un backbone basato su ResNet50, addestrato con tecniche di learning rate scheduling, per approfondire le implicazioni di un addestramento esteso.

I risultati sperimentali evidenziano che, sebbene l'approccio YOLO si dimostri robusto nella gestione delle complessità dei dati SAR, l'integrazione di metodologie basate su modelli Transformer richiede ulteriori approfondimenti per una piena valorizzazione delle peculiarità radar. In conclusione, l'elaborato sottolinea la complementarietà tra approcci classici e innovativi, delineando prospettive future per l'ottimizzazione dei sistemi di object detection in ambito SAR, con potenziali applicazioni nell'osservazione della Terra e nel monitoraggio di infrastrutture e attività critiche.

Contents

Chapter 1

Introduction

1.1 Context

This thesis was carried out as part of an industrial internship at a company specializing in Synthetic Aperture Radar (SAR) technology **ICEYE** [8]. During the internship, I had the unique opportunity to work closely with experts in radar imaging and data processing, gaining access to an extensive repository of ultra-high-resolution SAR imagery. Such hands-on experience with real-world data made it possible to identify pressing challenges and research gaps in automated target recognition, particularly for aircraft detection tasks.

SAR imaging has become a key technology in Earth Observation (EO) due to its ability to capture detailed images regardless of weather conditions or lighting. In recent years, new satellite constellations have been launched, offering meter and in some cases 25cm resolution. These exceptionally high resolutions promise more detailed insights into surface features; however, the inherent radar-specific complexities (such as speckle noise, double-bounce reflections, and side-looking geometry) pose unique difficulties for automated object detection systems.

1.2 Research Rationale

Object Detection (OD) in SAR data has been an area of active exploration. While classic OD pipelines—often designed for optical images—can be adapted to radar data, many open questions persist regarding how well these models capture radar-

specific phenomena. In particular:

- **High resolution** in SAR imagery can amplify both signal details and radar noise/artifacts. It remains unclear whether these computationally expensive resolutions always translate into more accurate detections.
- **Colorized Sub-aperture Images (CSI)**, which encode angular scattering information, might offer a richer data representation but also introduce additional complexity.
- **Variable resolutions and geometric distortions** are typical of SAR datasets, complicating the choice of hyperparameters (such as patch size, stride, or tiling strategies) for robust detection.
- **Transfer learning approaches** from pre-trained models (often optical or multi-spectral) may reduce the expense of annotating SAR data, if adapted carefully to radar-specific characteristics.

Since the thesis was developed in an industrial setting, these challenges had direct practical relevance. The solutions explored aimed to balance high accuracy with real-world feasibility—both in terms of computational resources and annotation effort.

1.3 Research Questions

Based on the above considerations, the thesis addresses the following research questions:

1. **Effectiveness of standard OD architectures on SAR:**
   Can widely used Object Detection models (e.g., the YOLO family), originally tailored for optical imagery, be effectively applied to ultra-high-resolution SAR data?
2. **Utility of CSI representations:**
   Do Colorized Sub-aperture Images, which capture angular backscattering information, provide measurable improvements in detection performance over conventional amplitude-only SAR (GRD) images?

3. **Trade-off of high resolution:**
   Does the extremely high spatial resolution offered by some SAR systems justify its significant computational cost, or do more moderate resolutions suffice for reliable object detection?

4. **Optimal hyperparameters for varying resolutions:**
   How should tile/patch sizes and other preprocessing steps be optimized to handle SAR data that may exhibit considerable variability in resolution and geometry?

5. **Transformer-based models for variable input sizes:**
   Given that SAR images can differ in size and shape, can Vision Transformers (ViTs) or foundation models (like *Clay*)—which accept variable numbers of input tokens—offer a more flexible approach to SAR object detection?

6. **Data annotation efficiency via EO foundation models:**
   SAR data labeling typically requires specialized expertise. To what extent can an Earth Observation foundation model, pre-trained on large-scale datasets, reduce the reliance on extensive SAR-specific annotations?

1.4 Thesis Outline

This thesis is organized into six chapters, each addressing a different aspect of the research:

- **Chapter 2** - Background and Theoretical Foundations
  Reviews fundamental concepts of Synthetic Aperture Radar (SAR) imaging and provides an overview of object detection methods, with particular emphasis on deep learning architectures.

- **Chapter 3** - Environment and Hardware Setup
  Describes the computational environment, hardware resources, and software libraries utilized in this work.

- **Chapter 4** - Dataset Acquisition and Preprocessing
  Details the process of collecting SAR images, focusing on the modalities pro-

vided by the satellite constellation. It also covers data preparation strategies—such as calibration, tiling, and annotation—that are crucial for producing robust training datasets.

- **Chapter 5** - YOLO for Object Detection in SAR Imagery
  Explains how YOLO-based models, originally developed for optical images, were adapted to handle SAR-specific characteristics. Metrics, training protocols, and performance evaluations are discussed to highlight the strengths and limitations of this approach.

- **Chapter 6** - Foundation Model:Clay Enhancements for Object Detection
  Presents the integration of a Transformer-based foundation model, Clay, as an alternative or complementary method to classic convolutional architectures. This chapter illustrates how Clay's flexible token-based input can adapt to varying SAR resolutions, along with the fine-tuning strategies employed.

- **Chapter 7** - Conclusions and Future Work
  Summarizes the main findings of the thesis, discussing both practical and theoretical contributions. It also outlines potential directions for further research, particularly in leveraging foundation models for SAR applications.

Chapter 2

Background and Theoretical Foundations

2.1 Synthetic Aperture Radar (SAR) and SAR Images

Synthetic Aperture Radar (SAR) is one of the most important active remote sensing technologies available today. Unlike optical sensors, SAR uses microwave radar signals to "illuminate" the Earth's surface and record the backscattered energy, enabling imaging regardless of weather or lighting conditions [1]. This section introduces SAR, explains its operating principles, and discusses the main characteristics and challenges associated with SAR imagery.

2.1.1 Introduction to SAR

What is SAR?

Synthetic Aperture Radar is an active sensor system that transmits electromagnetic (EM) waves—typically in the microwave range—and records the echoes that return after these waves interact with the target surface. The key innovation behind SAR is its ability to simulate a large antenna aperture by moving the radar along a flight path (on an aircraft or satellite) and coherently processing the received echoes. This "synthetic" aperture greatly enhances the spatial resolution of the radar images, making SAR an indispensable tool for Earth Observation.

Significance in Remote Sensing:

SAR offers several critical advantages:

- **All-Weather and Day-Night Operation:** Because SAR relies on microwave radiation, its performance is largely independent of sunlight and can penetrate clouds, fog, and rain.
- **High Spatial Resolution:** By synthesizing a large aperture, SAR systems can achieve fine spatial resolutions that are comparable to or even exceed those of optical sensors in certain conditions.
- **Versatility in Applications:** SAR is used in diverse applications such as environmental monitoring, disaster management, military reconnaissance, and mapping of urban and rural areas.

Basic Operating Principles:

At its core, a SAR system emits pulses of microwave energy towards the Earth and measures the energy that is reflected back. The time delay between the transmitted pulse and the received echo is used to determine the distance (or slant range) to the target. By moving along a predefined trajectory and processing multiple echoes coherently, SAR synthesizes a large virtual aperture, which allows it to resolve details that would be impossible with a physically small antenna. The resulting data, often in the form of complex numbers (amplitude and phase), are processed to generate images that represent the backscatter intensity of the target area.

2.1.2 SAR Image Principles, Characteristics, and Challenges

This subsection delves into the specifics of how SAR images are formed, the inherent characteristics of these images, and the challenges that arise during acquisition and interpretation.

SAR Image Formation

Acquisition Process:

SAR image formation begins with the emission of microwave pulses, which are modulated (or "chirped") to improve range resolution. As the radar platform moves, echoes from different parts of the target are received at slightly different times.

These echoes are then coherently integrated using sophisticated signal processing algorithms. The result is a two-dimensional "image" where each pixel's intensity corresponds to the strength of the returned signal (or backscatter) from a particular area on the Earth's surface.

Key Characteristics of SAR Images:

1. **Side-Looking Geometry and Slant Range:**
   Unlike optical sensors that typically capture images looking straight down (nadir view), SAR systems are inherently side-looking. This configuration means that the distance from the sensor to the target—the slant range—differs from the actual horizontal (ground) distance (Figure 2.1).

   - **Implications:**
     - *Geometric Distortions:* Objects can appear shifted, foreshortened, or even layover (when tall objects cause overlap in the image) due to the angle of observation (Figure 2.2).
     - *Resolution Considerations:* The effective resolution depends on both the transmitted signal's characteristics and the viewing geometry.

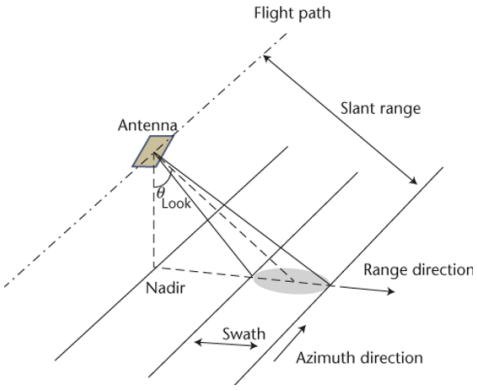


Figure 2.1: SAR Geometry with its peculiar Side-Looking Geometry.

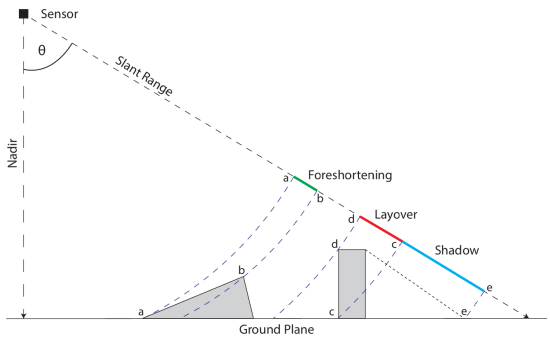


Figure 2.2: Slant Range and types of Geometric Distortions.

2. **Double Bouncing Phenomenon:**
   SAR images sometimes exhibit the "double bounce" effect, where a radar pulse reflects off a flat surface (like the ground) and then off a vertical object (such as a building or tree) before returning to the sensor (Figure 2.3).

- **Implications:**

  Ambiguous Representation: This effect can cause the target to appear twice or at an incorrect location in the image, complicating interpretation and detection tasks.

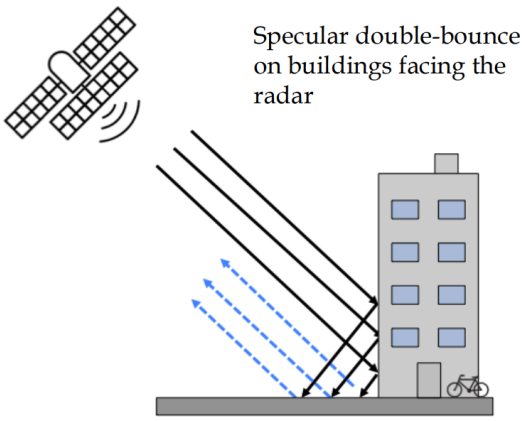


Figure 2.3: Double Bouncing

3. **Bandwidth and Resolution Trade-Offs:**

   The resolution of a SAR image is directly influenced by the bandwidth of the transmitted signal. A wider bandwidth allows for finer resolution because it can distinguish between targets that are closely spaced in range.

   - **Implications:**
     - *Signal-to-Noise Ratio (SNR):* Increasing bandwidth may improve resolution but also influences noise characteristics and penetration capabilities.
     - *Design Considerations:* Engineers must balance bandwidth to optimize both resolution and overall image quality.

4. **Speckle Noise and Other Artifacts:**

   Due to the coherent nature of SAR signal processing, images are typically corrupted by speckle noise—a granular interference pattern that arises from the constructive and destructive interference of the returned radar signals.

   - **Implications:**
     - *Image Degradation:* Speckle noise can obscure fine details and make automated image interpretation more challenging.

  - *Preprocessing Needs:* Specialized filtering and calibration techniques are required to reduce speckle while preserving important features.

Electromagnetic Spectrum and Radar Bands:

SAR sensors operate in the microwave region of the electromagnetic (EM) spectrum, which typically ranges from about 1 mm to 1 m in wavelength (Figure 2.4). The choice of operating frequency (or band) greatly influences the imaging characteristics:

- **L-Band (1–2 GHz):**
  - *Longer Wavelengths:* Offers better penetration through vegetation and soil, making it suitable for forest monitoring and soil moisture studies.
  - *Lower Resolution:* Due to the longer wavelength, L-band sensors generally provide coarser spatial resolution compared to higher-frequency bands.
- **C-Band (4–8 GHz):**
  - *Moderate Wavelengths:* Widely used for weather radar and Earth observation, C-band strikes a balance between resolution and penetration capability.
  - *Sensitivity:* Less penetration than L-band but provides improved resolution, making it useful for urban mapping and agricultural monitoring.
- **X-Band (8–12 GHz):**
  - *Shorter Wavelengths:* Provides the highest spatial resolution among these bands, making it ideal for detailed structural mapping (e.g., urban areas).
  - *Lower Penetration:* X-band waves are more sensitive to surface features and are more affected by atmospheric conditions such as rain, which can attenuate the signal.

These differences in operating frequency lead to distinct advantages and trade-offs. For example, while L-band SAR may penetrate through tree canopies and snow, X-band SAR offers superior detail for mapping buildings and other small features.

Understanding these spectral characteristics is essential when selecting the appropriate SAR data for a specific remote sensing application (Figure 2.5).

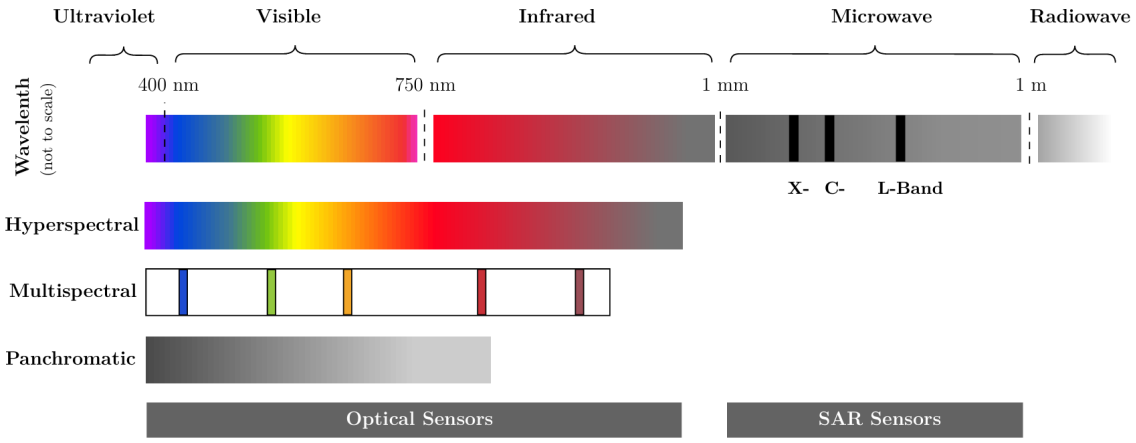


Figure 2.4: Electromagnetic Spectrum with .

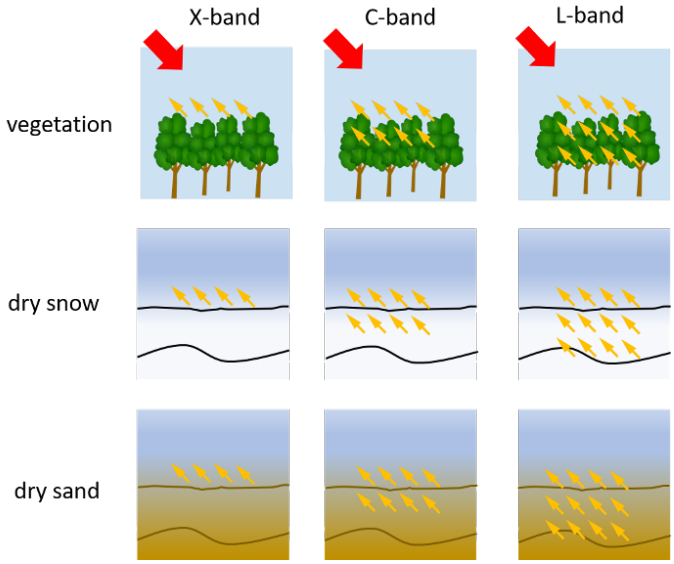


Figure 2.5: Penetration comparison between X-band, C-band and L-band.

Challenges in Interpreting and Processing SAR Data:

SAR images, while rich in information, present unique challenges compared to optical images:

- **Geometric Distortions:**
  - *Foreshortening:* Due to side-looking geometry, parts of the terrain may appear compressed.

  - *Layover:* Tall objects may appear displaced from their actual location, causing overlaps in the image.

- **Radiometric Calibration:**

  - *Backscatter Variability:* Converting raw digital numbers (DNs) to physically meaningful backscatter values requires complex calibration, accounting for factors like antenna gain, incidence angle, and system noise.

- **Complex Data Nature:**

  - *Amplitude and Phase Information:* SAR data typically contain both amplitude and phase information. While amplitude is directly related to the intensity of backscatter, phase can provide additional information (such as elevation in interferometric SAR) but is also more susceptible to noise and decorrelation.

- **Processing Requirements:**

  - *Advanced Algorithms:* Effective SAR image processing necessitates robust algorithms for image formation, noise reduction, geometric correction, and calibration.

  - *Computational Load:* The high data rates and the complexity of SAR signal processing can require significant computational resources.

2.2 ICEYE Oy

ICEYE [8] operates a fast-growing constellation of microsatellites equipped with X-band Synthetic Aperture Radar (SAR) sensors, deployed in Low Earth Orbit (LEO) at altitudes generally between 500 km and 600 km. As of recent updates, 44 satellites since 2018 (as of January 2025), with more than 20 new satellites planned annually for 2025 and beyond. Each satellite has a compact design—often weighing around 100 kg—and integrates high-performance electronics to capture high-resolution radar images in varied weather conditions.
Key features of the ICEYE constellation include:

- **High Revisit Frequency** By distributing satellites across multiple orbital planes, ICEYE achieves frequent revisit times—often with the ability to image a given location multiple times per day. This near-real-time imaging capability is crucial for disaster management, maritime surveillance, and any application requiring timely situational awareness.

- **Global Coverage** The multi-orbit approach ensures that the constellation collectively covers all regions of the globe, providing consistent imaging opportunities from the poles to the equator. The network design allows for broad scanning modes that can capture vast swaths of terrain, or narrower spotlight modes delivering fine detail.

- **Agility and Responsiveness** The relatively small size of each microsatellite makes it easier to maneuver and re-task imaging priorities. This rapid retargeting ensures that high-interest sites—such as natural disaster zones or strategic areas—can be revisited quickly, often within hours.

- **Flexible Growth and Enhancement** ICEYE continues to add satellites to its constellation, further boosting temporal and spatial resolution. Each new deployment refines the overall imaging cadence and coverage. Moreover, advanced miniaturization technology allows payloads to incorporate improvements in antenna design and onboard processing without significantly increasing satellite mass.

- **Technical Highlights**

  - **X-Band SAR Operation:** Typically centered near 9.65 GHz, offering high-resolution imaging with various bandwidth options (300, 600, or 1200 MHz) for different acquisition modes.

  - **Data Latency:** Ground stations placed worldwide enable rapid data downlink, minimizing the time from image acquisition to final product delivery.

  - **Resolution and Swath:** Depending on acquisition mode (e.g., Stripmap, Dwell, Spotlight), ground resolutions can reach sub-meter levels, while swath sizes range from a few kilometers to broader coverage scenarios.

2.2.1 Acquisition Modes

ICEYE's SAR imaging capabilities are enabled by a set of specialized acquisition modes, each tailored to balance the trade-offs between resolution, swath width, and revisit frequency. These modes are based on variations in radar bandwidth and beam steering techniques [11]. The following describes each mode in detail:

- **Spot Fine** Spot Fine mode actively steering the radar beam to continuously track a specific target area during the acquisition that lasts for 15 seconds (Figure 2.6). The dynamic beam steering significantly increases the effective synthetic aperture in the azimuth direction, thereby dramatically improving azimuth resolution. Using a bandwidth of 600 MHz, Spot Fine mode achieves high-resolution imagery—often nearing 0.5 m or better—over a smaller, focused area. This mode is ideal when pinpointing fine details of a critical target is paramount, even at the expense of reduced swath width.

- **DWELL** In standard DWELL mode, the radar remains fixed on a target area for the full 25-second duration. With a typical bandwidth of around 300 MHz, the collected data is processed into sub-apertures that capture the temporal evolution of the backscatter. This mode produces a moderate-resolution image—typically around 1.0 m ground resolution—over a scene approximately 5 km $\times$ 5 km. The extended integration time improves the signal-to-noise ratio (SNR), making it well-suited for applications requiring reliable detection under varied conditions.

- **DWELL Fine** DWELL Fine mode enhances the standard DWELL approach by utilizing an increased transmitted bandwidth, usually around 600 MHz or higher. The wider bandwidth results in finer range resolution, enabling the detection and discrimination of smaller or closely spaced objects. The acquisition procedure remains 25 seconds but the improved resolution (approximately 0.5 ground resolution) makes this mode ideal for detailed analysis in urban environments or infrastructure monitoring.

- **Dwell Precise** Dwell Precise mode represents a further refinement of the DWELL approach. It employs an even higher radar bandwidth—up to 1200

MHz—combined with enhanced calibration and processing techniques to achieve ultra-high resolution imagery. In Dwell Precise mode the increased bandwidth and rigorous processing yield ground resolutions approaching 0.25 m (25 cm). This mode is particularly valuable for engineering tests and applications where extremely fine spatial details are critical, such as precision mapping, infrastructure monitoring, and detailed change detection.

Table 2.1: Acquisition Mode comparison

Mode	**Acquisition**	**Ground resolution**	**CSI**	**Nominal swath width x length (km)**	**Collection duration (s)**
Dwell Precise	1200 MHz	25 cm	Yes	5 × 5	25
Dwell Fine	600 MHz	50 cm	Yes	5 × 5	25
Dwell	300 MHz	1 m	Yes	5 × 5	25
Spot Fine	600 MHz	50 cm	No	5 × 5	15

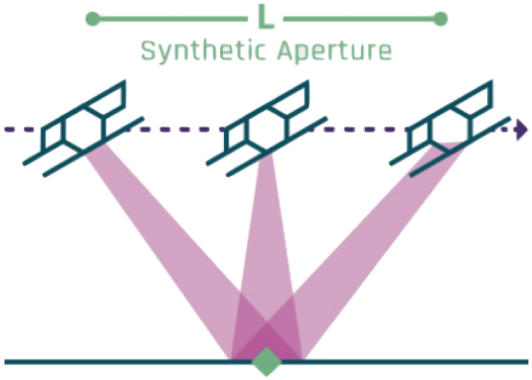


Figure 2.6: Spotlight Mode.

2.2.2 GRD (Ground Range Detected)

The GRD product is derived from the complex SAR data through a process known as "detection," where the in-phase and quadrature components (representing amplitude and phase) are converted into amplitude-only values [9]. This process yields a simple amplitude image that retains the native sensor geometry, meaning that the data are presented in a range-azimuth layout—the same orientation as acquired by the radar system. In contrast to map-projected images, this native format avoids interpolation artifacts that can obscure fine details and affect geolocation accuracy. After

detection, the amplitude data are usually multi-look processed to reduce speckle noise, and then projected onto the ground plane using an Earth ellipsoid model (typically the WGS84 reference ellipsoid, with a semi-major axis of 6,378,137.0 m and a semi-minor axis of 6,356,752.314245 m). A fixed, averaged terrain height is applied to bring the ellipsoid surface closer to the true ground surface. The resulting GRD images feature equidistant pixel spacing in both range and azimuth, offering a near-circular spatial resolution. Furthermore, GRD products are stored in a GeoTIFF container using unsigned 16-bit integer representation. This format not only ensures compatibility with standard image processing and GIS tools (such as QGIS or GDAL) but also embeds comprehensive metadata—including ground control points (GCPs) and rapid positioning capability (RPC) coefficients—to support precise geospatial exploitation and rigorous geolocation. Although the term "GRD" originates from legacy engineering jargon, it remains widely used within the SAR community to denote amplitude images in their native sensor orientation.

2.2.3 CSI: Colorized Sub-aperture Image

The CSI (Colorized Sub-aperture Image) technique is an innovative method specifically implemented during Dwell mode acquisitions to enhance the interpretability of SAR data [10]. Unlike conventional SAR images that produce a single grayscale representation by coherently processing the entire synthetic aperture, CSI leverages the temporal diversity within a Dwell collection by dividing the total acquisition time into multiple sub-apertures (Figure 2.7). Here's how the process works in detail:

- **Dwell Mode Collection and Sub-aperture Division** In Dwell mode, the total collection time (typically 25 seconds) is partitioned into 13 discrete sub-apertures, each lasting approximately 1.91 seconds. This segmentation captures slightly different radar returns within the same overall acquisition, with each sub-aperture corresponding to a unique portion of the Doppler spectrum.

- **Individual Sub-aperture Processing and Color Assignment** Each sub-aperture is processed independently to extract its backscatter data. The key innovation in CSI is the assignment of a unique color to each sub-aperture based on its acquisition sequence: the earliest sub-aperture is assigned red,

and the colors gradually shift through the spectrum—transitioning from red, through orange, yellow, green, and finally to blue for the final sub-aperture. This color coding represents the predominant scattering direction of a point in the image over time.

- **Composite Image Formation** After processing, the individually colored sub-aperture images are merged into a single composite GeoTIFF product. In the final CSI image, the color information reveals how a target's scattering properties vary during the acquisition:
  - **Isotropic Scatterers** For objects that scatter equally in all directions (such as grass, trees, or water), the contributions from each sub-aperture are similar. When these are combined, the result is a near-neutral, grayscale appearance, akin to standard SAR images.
  - **Anisotropic Scatterers** Conversely, objects with dominant scattering directions—often human-made structures with flat surfaces or sharp angles—retain a strong color signature in the composite. These features stand out clearly against the more neutral background, drawing attention to built-up or structurally distinct elements.
- **Metadata and Data Accessibility** All pertinent information regarding each sub-aperture (such as the exact duration, assigned color in normalized RGB, mid-time of acquisition, satellite position, and velocity) is embedded within the GeoTIFF metadata. This detailed metadata can be accessed using tools like GDAL or QGIS, enabling users to precisely interpret the CSI product and correlate the color information with physical scattering characteristics.

2.3 Object Detection Fundamentals

Object detection is a core task in computer vision that extends beyond mere image classification. It involves not only identifying what objects are present in an image but also precisely localizing them by drawing bounding boxes around them. This chapter provides a comprehensive overview of object detection, outlining its defini-

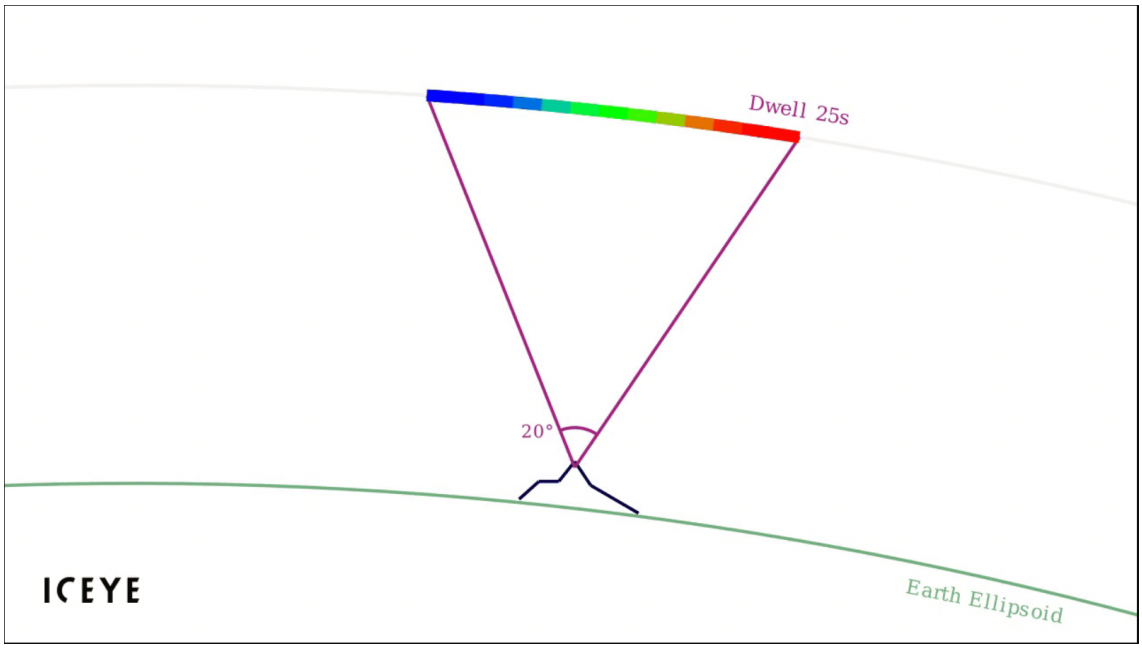


Figure 2.7: Detail of a Dwell acquisition and how each color is assigned when forming a Colorized Sub-aperture Image (CSI).

tion, importance, and various architectures, as well as the key evaluation metrics used to assess detector performance.

2.3.1 What is Object Detection?

Object detection is the process of automatically identifying and localizing objects within an image or a video frame. Unlike image classification, which assigns a single label to an entire image, object detection requires the system to pinpoint the location of one or more objects by predicting their positions and sizes through bounding boxes.

Role in Computer Vision:

- **Scene Understanding:** Object detection helps computers gain a deeper understanding of scenes by identifying multiple objects and their spatial relationships.

- **Real-Time Applications:** It is critical for real-time systems such as autonomous driving, video surveillance, robotics, and remote sensing, where quick and accurate decisions are necessary.

- **Remote Sensing:** In remote sensing applications, object detection is used to identify and monitor features such as vehicles, buildings, or even specific targets within satellite or SAR imagery.

Why It Is Important:

- **Automation:** Automating object detection reduces the need for manual annotation and enables large-scale analysis of visual data.
- **Enhanced Decision-Making:** By providing detailed information about object locations and identities, detection systems support decision-making processes in various fields, including security, urban planning, and environmental monitoring.
- **Integration in Advanced Systems:** Object detection is a building block for higher-level tasks such as object tracking, scene segmentation, and activity recognition.

2.3.2 Detection Architectures: One-Stage vs. Two-Stage

Object detection architectures can generally be categorized into two groups based on their processing pipelines: one-stage detectors and two-stage detectors.

One-Stage Detectors

One-stage detectors [14], such as YOLO [15] (You Only Look Once) and SSD (Single Shot MultiBox Detector), approach object detection as a regression problem. They process the entire image in a single pass through the network and simultaneously predict bounding boxes and class probabilities for each grid cell.

- **Key Characteristics:**
  - **Speed:** Because they use a single network for detection, one-stage detectors are extremely fast, often capable of real-time performance.
  - **Global Context:** These methods process the whole image at once, allowing the model to leverage global context when making predictions.
  - **Simplicity:** The unified architecture simplifies the training and inference pipeline.

- **Pros:**
  - High efficiency and low latency, ideal for real-time applications.
  - Lower computational complexity compared to multi-stage methods.
- **Cons:**
  - They may sacrifice some accuracy, particularly in detecting small objects or objects that are close together, because of the coarse division of the image into grids.

Two-Stage Detectors

Two-stage detectors, such as Faster R-CNN and R-FCN, first generate region proposals that likely contain objects and then perform classification and refinement of these proposals in a second stage.

- **Key Characteristics:**
  - **Region Proposals:** The first stage uses algorithms like Selective Search or a Region Proposal Network (RPN) to propose candidate object regions.
  - **Refinement Stage:** The second stage refines these proposals, classifying each one and adjusting the bounding box coordinates for greater accuracy.
- **Pros:**
  - Generally higher accuracy and better localization precision, especially for small objects.
  - More robust to cluttered scenes due to the refinement stage that focuses on high-quality region proposals.
- **Cons:**
  - Slower inference times due to the sequential processing steps.
  - More complex training and higher computational requirements, which may limit real-time applicability.

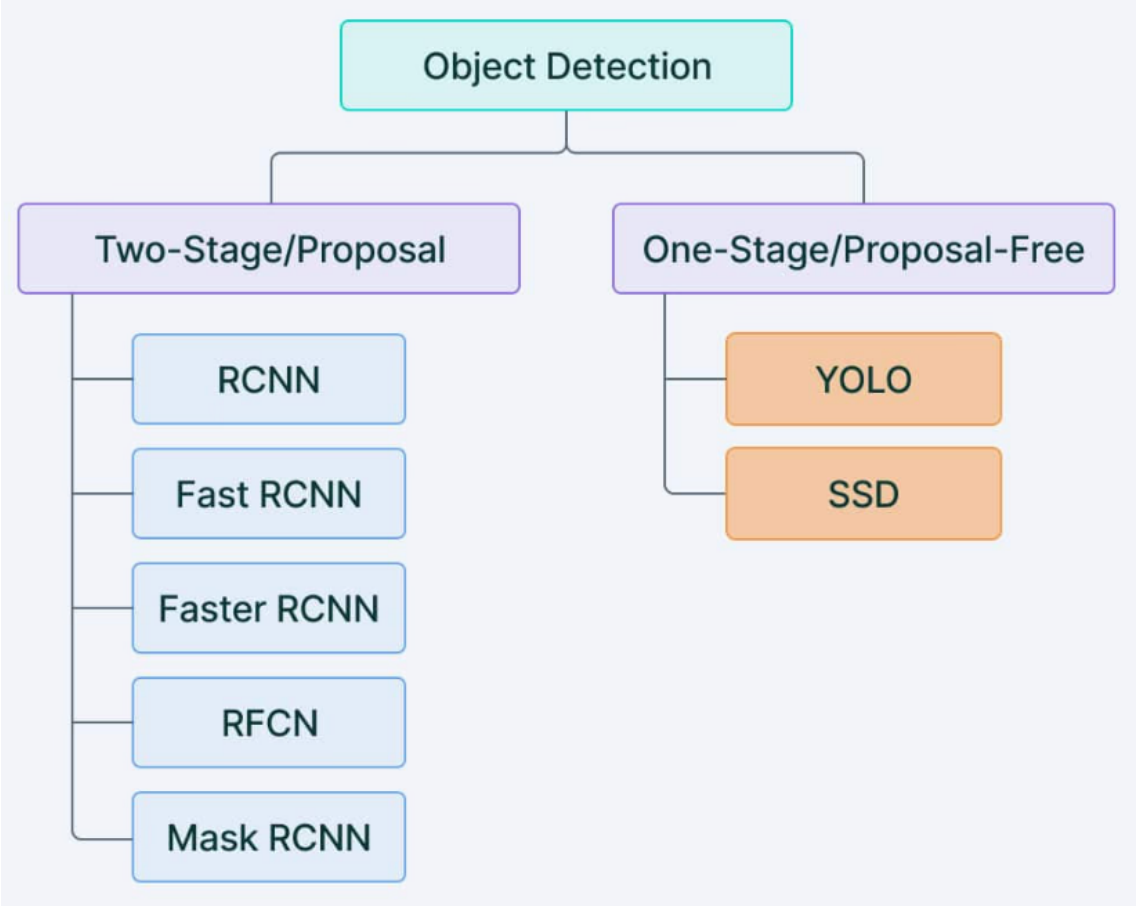


Figure 2.8: One and two stage detectors.

2.3.3 Metrics for Object Detection

Evaluating object detection systems involves measuring both the localization accuracy of bounding boxes and the classification accuracy of detected objects. Several metrics are used to assess performance:

Intersection over Union (IoU)

Intersection over Union (IoU) is a widely used metric in object detection for assessing the accuracy of predicted bounding boxes. It is defined as the ratio of the area of overlap between the predicted bounding box B_p and the ground truth bounding box B_{gt} to the area of their union:

$$\text{IoU} = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|}$$

This measure provides a simple yet effective quantification of how well the prediction matches the ground truth. A higher IoU value indicates a better overlap between the boxes, and typically, an IoU threshold of 0.5 is used to determine if a detection qualifies as a true positive. IoU is crucial because it penalizes inaccuracies in positioning and size, ensuring that only precise detections are rewarded (Figure 2.9).

Complete Intersection over Union (CIoU)

While IoU focuses solely on the area of overlap, Complete Intersection over Union (CIoU) extends this concept by considering additional geometric factors that affect bounding box quality. CIoU takes into account the distance between the centers of the predicted and ground truth boxes, as well as the similarity of their aspect ratios. The CIoU loss is mathematically formulated as:

$$\text{CIoU} = \text{IoU} - \frac{\rho^2(b, b_{gt})}{c^2} - \alpha v$$

In this equation:

- $\rho(b, b_{gt})$ denotes the Euclidean distance between the centers of the predicted box b and the ground truth box b_{gt}.
- c represents the diagonal length of the smallest enclosing box that covers both b and b_{gt}.
- v measures the consistency of the aspect ratios between the two boxes.

$$v = \frac{4}{\pi^2}\left(\arctan\left(\frac{w^{gt}}{h^{gt}}\right) - \arctan\left(\frac{w}{h}\right)\right)^2,$$

- α is a positive trade-off parameter that balances the influence of the aspect ratio term.

$$\alpha = \frac{v}{(1 - \text{IoU}) + v}$$

By integrating these components, CIoU provides a more comprehensive assessment of localization accuracy. It penalizes not only poor overlap but also misalignment and discrepancies in shape, making it particularly effective as a loss function during model training for improved bounding box regression (Figure 2.9).

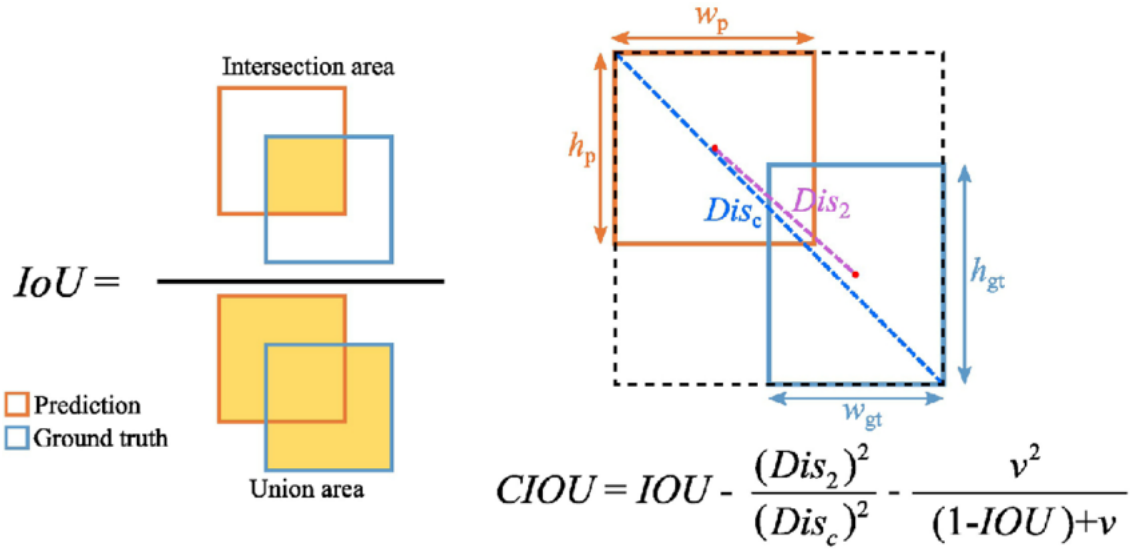


Figure 2.9: Illustrations of the intersection over union (IoU) and complete intersection over union (CIoU).

Average Precision (AP)

Average Precision (AP) is a widely used metric for evaluating the performance of object detection models on a per-class basis. It is calculated by measuring the area under the precision-recall curve for a specific object class. In more formal terms, precision P and recall R are defined as:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad R = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where TP (true positives) represents correctly identified objects, FP (false positives) represents incorrect detections, and FN (false negatives) indicates missed objects. The precision-recall curve is obtained by plotting precision against recall at various confidence thresholds. The area under this curve is the Average Precision for that

particular class. Conceptually, AP provides a single numeric value that summarizes how well a model balances precision and recall. A higher AP indicates that the model consistently maintains both high precision and high recall across different thresholds, effectively capturing the trade-off between detecting as many objects as possible (recall) and avoiding incorrect detections (precision). Consequently, AP is an essential metric for comparing how different models perform on a single class of objects.

Mean Average Precision (mAP)

While AP measures the performance for one specific class, Mean Average Precision (mAP) extends this idea to multiple classes. It is defined as the mean of the AP values computed for all object classes:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i,$$

where N is the total number of classes, and AP_i is the Average Precision for the i-th class. By taking the average across all classes, mAP serves as a comprehensive measure of an object detector's overall performance. It is widely used in benchmark challenges, such as the COCO and PASCAL VOC datasets, allowing researchers and practitioners to compare models on a level playing field.

Other Relevant Metrics

In addition to AP and mAP, several other metrics are commonly employed to provide a more complete picture of a detector's capabilities:

Precision and Recall. Precision and recall form the basis for most object detection metrics. Precision focuses on the fraction of correct detections among all detections made by the model, whereas recall measures the fraction of actual objects that the model successfully identifies. Balancing these two aspects is crucial for real-world scenarios.

F1 Score. The F1 score is defined as the harmonic mean of precision and recall:

$$F_1 = 2 \times \frac{P \times R}{P + R}.$$

It offers a single metric that combines both precision and recall, providing insight into how well the model handles the trade-off between false positives and missed detections.

Inference Time. In many practical applications, especially those requiring real-time responses (e.g., autonomous vehicles or surveillance), the time required to process an image can be just as important as accuracy. Models with higher accuracy but excessive inference times may be impractical in latency-sensitive environments.

False Positive Rate. False positive rate indicates how often the model incorrectly labels non-object regions as objects. It is particularly important in cluttered or complex scenes, where the cost of an erroneous detection can be significant. Monitoring this rate helps ensure that the model remains robust even when faced with challenging backgrounds or overlapping objects.

2.4 The YOLO Family of Object Detectors

2.4.1 Overview of YOLO

YOLO [15], which stands for "You Only Look Once", revolutionized object detection by reframing it as a single regression problem. Unlike traditional two-stage methods that first generate region proposals and then classify them, YOLO uses a unified, end-to-end neural network that processes the entire image in one forward pass. In the YOLO framework, the input image is divided into an S $\times$ S grid. Each grid cell is responsible for predicting a fixed number (B) of bounding boxes, along with a confidence score and conditional class probabilities for the object it contains. The confidence score reflects both the likelihood that an object is present and the accuracy of the predicted bounding box (typically measured by the Intersection over Union, or IoU). The key benefits of this unified detection pipeline are its simplicity and speed. By leveraging a single neural network to perform both localization and classification simultaneously, YOLO achieves real-time performance with low latency, making it ideal for applications such as video surveillance, autonomous driving, and, as in our work, remote sensing. Furthermore, YOLO's global reasoning—processing the entire image at once—helps reduce false positives that can occur when only local regions are considered. The network is trained end-to-end using a composite loss function that balances errors in bounding box regression, confidence prediction, and class probability estimation. (For additional insights on the basic

mechanism of YOLO, see .)

2.4.2 Evolution of YOLO Versions

Since its introduction in 2015, YOLO has evolved through several iterations, each building on the success of its predecessor while addressing its limitations:

YOLOv1

The original YOLO model demonstrated that a single convolutional network could directly predict bounding boxes and class probabilities from full images. Despite its remarkable speed (processing images at approximately 45 frames per second), YOLOv1 struggled with localizing small objects and handling multiple objects within the same grid cell [15].

YOLOv2 (YOLO9000)

An important upgrade, YOLOv2 introduced the use of anchor boxes, which allowed the network to better predict bounding boxes for objects of varying shapes and sizes. It also implemented batch normalization and a multi-scale training strategy. These improvements not only enhanced accuracy—achieving state-of-the-art performance on datasets like PASCAL VOC and COCO—but also maintained real-time speeds. (For a more detailed discussion on YOLOv2's improvements, see [12].)

YOLOv3

YOLOv3 further refined the architecture by adopting Darknet-53 as its backbone—a deeper and more robust network that leverages residual connections. It employs feature pyramid networks to predict objects at multiple scales, improving its performance on small objects. YOLOv3 also enhanced the loss function by combining classification and localization losses more effectively, resulting in a balanced trade-off between speed and accuracy. (Refer to for additional technical details on YOLOv3 [13].)

YOLOv4 and YOLOv5

These versions focused on optimizing both efficiency and detection performance. YOLOv4 [2] introduced techniques such as Spatial Pyramid Pooling (SPP) and Cross-Stage Partial Networks (CSP), which improved feature extraction and generalization. YOLOv5, maintained as an open-source project by Ultralytics, offered a more modular and user-friendly framework. It brought improvements in model scal-

ability, dynamic anchor box generation, and more efficient training routines, making it adaptable across various computational budgets.

YOLOv6 and YOLOv7

The subsequent versions continued to refine the model by incorporating lightweight backbones (such as EfficientNet variants), further optimizing anchor box strategies, and integrating advanced loss functions like focal loss to better handle class imbalance and improve small object detection. These iterations maintained the YOLO family's reputation for real-time performance while incrementally boosting accuracy [4] [3].

YOLOv8

The latest advancement, YOLOv8 [5], synthesizes lessons learned from all previous iterations. It integrates a more powerful backbone with improved feature extraction capabilities, a refined neck for multi-scale feature fusion, and a modernized detection head that, in some configurations, adopts an anchor-free mechanism. These changes collectively enhance both localization precision and classification robustness, positioning YOLOv8 as one of the most efficient and accurate real-time object detectors available today (Figure 2.10).

Figure 2.10: Timeline of You Only Look Once (YOLO) variants.

2.4.3 YOLOv8 Architecture

YOLOv8 represents the cutting edge of the YOLO series, incorporating several key architectural improvements over earlier versions [20]:

- **Enhanced Backbone:** YOLOv8 utilizes an advanced backbone network designed to extract richer and more discriminative features from the input image. This backbone often integrates recent innovations in convolutional neural networks (CNNs), such as improved residual blocks and attention mechanisms, to capture both local details and global context effectively.

- **Optimized Neck and Feature Aggregation:** The neck of YOLOv8 is responsible for fusing features from different layers of the backbone. It uses a combination of upsampling, concatenation, and additional convolutional layers to generate a multi-scale feature pyramid. This design allows the model to detect objects of various sizes more reliably and improves its overall detection performance, particularly in challenging conditions such as those encountered in SAR imagery.

- **Modern Detection Head:** In YOLOv8, the detection head has been revamped to predict bounding box coordinates, objectness scores, and class probabilities more accurately. Some configurations of YOLOv8 adopt an anchor-free approach, which removes the dependency on manually defined anchor boxes. Instead, the network directly regresses bounding box parameters, thereby simplifying the training process and improving flexibility across different object scales and aspect ratios.

- **Refined Loss Function and Training Strategies:** YOLOv8 employs a composite loss function that carefully balances localization, classification, and objectness errors. Innovations such as improved IoU-based loss components and dynamic weighting schemes help to mitigate issues related to small object detection and class imbalance. Additionally, modern data augmentation techniques and adaptive learning rate schedules contribute to more robust and stable training.

- **Performance and Efficiency:** The cumulative effect of these architectural enhancements is a model that not only achieves state-of-the-art accuracy on benchmark datasets but also retains the real-time inference speed that has been a hallmark of the YOLO family. YOLOv8 is designed to be scalable and deployable on a variety of hardware platforms, from high-end GPUs to edge devices.

2.5 Deep Learning Architectures for Object Detection

Deep learning has dramatically transformed the field of computer vision, particularly for object detection tasks. Modern detectors rely on powerful neural network architectures that can automatically learn hierarchical features from raw data. This chapter focuses on three key areas: Convolutional Neural Networks (CNNs) [16], various types of convolution operations, and Vision Transformers (ViT). Each section provides an overview of the underlying principles, operational mechanisms, and how these architectures contribute to state-of-the-art object detection.

2.5.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are the cornerstone of modern object detection. They are designed to process data with grid-like structures (e.g., images) by learning hierarchical feature representations. CNNs exploit the local connectivity and stationarity of image data to significantly reduce the number of parameters compared to fully connected architectures.

Fundamental Building Blocks

1. **Convolutional Layers:**

   The key operation in a CNN is convolution. Given an input image (or feature map) X and a kernel K, a standard convolution operation produces an output feature map Y as follows:

$$Y(i,j) = \sum_{c=1}^{C_{\text{in}}} \sum_{m} \sum_{n} X_c(i+m, j+n) \cdot K_c(m,n) + b,$$

where:

- X_c is the cth channel of the input,
- K_c is the corresponding filter for channel c,
- b is a bias term, and
- The summations over m and n cover the spatial dimensions of the kernel.

2. **Locality Principle and Stationarity:**
   Convolutions are based on the assumption that local patterns (e.g., edges or textures) are more important than distant ones (locality) and that the statistics of the image do not change significantly across spatial locations (stationarity). These principles allow the same kernel to be applied across the entire image, dramatically reducing the number of parameters.

3. **Activation Functions:**
   After convolution, a nonlinear activation function is applied to introduce nonlinearity into the network. A common choice is the Rectified Linear Unit (ReLU), defined as:
   $$f(x) = \max(0, x).$$
   This activation function helps the network learn complex patterns by enabling it to model non-linear relationships.

4. **Pooling Layers:**
   Pooling layers perform downsampling to reduce the spatial dimensions of feature maps, thereby lowering computational cost and providing a degree of translation invariance. Two popular pooling strategies are:

   - **Max Pooling:** Takes the maximum value within a sliding window.
   - **Average Pooling:** Computes the average value within a sliding window.

5. **Max Pooling Operation:**
   If S is the pooling window size, the output $P(i, j)$ in max pooling is given by:
   $$P(i, j) = \max_{(m,n)\in S} X(i + m, j + n).$$

6. **Fully Connected Layers:**

   At later stages of the network, fully connected (FC) layers are used to perform high-level reasoning. FC layers flatten the multidimensional feature maps into a single vector and apply a learned linear transformation:

$$z = W \cdot \text{vec}(X) + b,$$

   where W is the weight matrix, b is the bias, and $\text{vec}(X)$ denotes the vectorized input. The FC layers often lead to the final classification or regression outputs.

7. **Loss Functions and Optimization:**

   During training, a loss function quantifies the discrepancy between the network's predictions and the ground truth. Common loss functions include:

   - **Cross-Entropy Loss:** Often used for classification tasks.
   - **Mean Squared Error (MSE):** Frequently used for regression tasks such as bounding box coordinate prediction in object detection.

   The network parameters are optimized using algorithms like Stochastic Gradient Descent (SGD) or adaptive methods (e.g., Adam), which adjust weights iteratively based on the computed gradients from backpropagation.

2.5.2 Convolution Types: Standard, Pointwise, and Depthwise

Modern CNN architectures enhance efficiency and flexibility by employing various types of convolutions. Here we detail three primary convolution types, providing general mathematical formulations and highlighting their benefits.

Standard Convolution

Standard convolution is the fundamental operation in convolutional neural networks (CNNs). Given an input image or feature map X with C_{in} channels and a convolutional kernel K of spatial size $k \times k$, the operation for each output feature map Y is defined as:

$$Y(i,j) = \sum_{c=1}^{C_{\text{in}}} \sum_{m=1}^{k} \sum_{n=1}^{k} X_c(i+m, j+n) \cdot K_c(m,n) + b.$$

In this formula, X_c denotes the cth channel of the input, K_c is the corresponding filter for that channel, and b represents a bias term. The summations over m and n run over the spatial dimensions of the kernel. This operation captures both the local spatial patterns and the relationships across channels, providing a robust mechanism for feature extraction. However, standard convolution can be computationally intensive when either the number of input channels C_{in} or the kernel size k is large, leading to a significant increase in the number of parameters.

Pointwise Convolution

Pointwise convolution simplifies the process of feature extraction by using a 1×1 kernel. Instead of considering a spatial neighborhood, pointwise convolution operates at each spatial location independently to combine information across the channels. Formally, it is expressed as:

$$Y(i,j) = \sum_{c=1}^{C_{\text{in}}} X_c(i,j) \cdot w(c) + b,$$

where $w(c)$ represents the weight associated with the cth channel. This type of convolution is especially useful for dimensionality reduction and channel fusion since it reduces the number of channels without changing the spatial dimensions. Because it requires far fewer parameters compared to standard convolution, pointwise convolution is frequently used in bottleneck layers and in architectures such as MobileNet, where computational efficiency is critical.

Depthwise Convolution

Depthwise convolution takes a different approach by processing each input channel independently with its own spatial kernel. For a given channel c, the operation is defined as:

$$Y_c(i,j) = \sum_{m=1}^{k} \sum_{n=1}^{k} X_c(i+m, j+n) \cdot K_c(m,n) + b_c,$$

where b_c is the bias term for the cth channel. By convolving each channel separately, depthwise convolution greatly reduces the number of parameters and the computational load, which is particularly beneficial in resource-constrained environments such as mobile devices. Often, depthwise convolution is followed by a pointwise convolution (a process known as depthwise separable convolution) to fuse the outputs from each channel, achieving a favorable balance between computational cost and model performance (Figure 2.11).

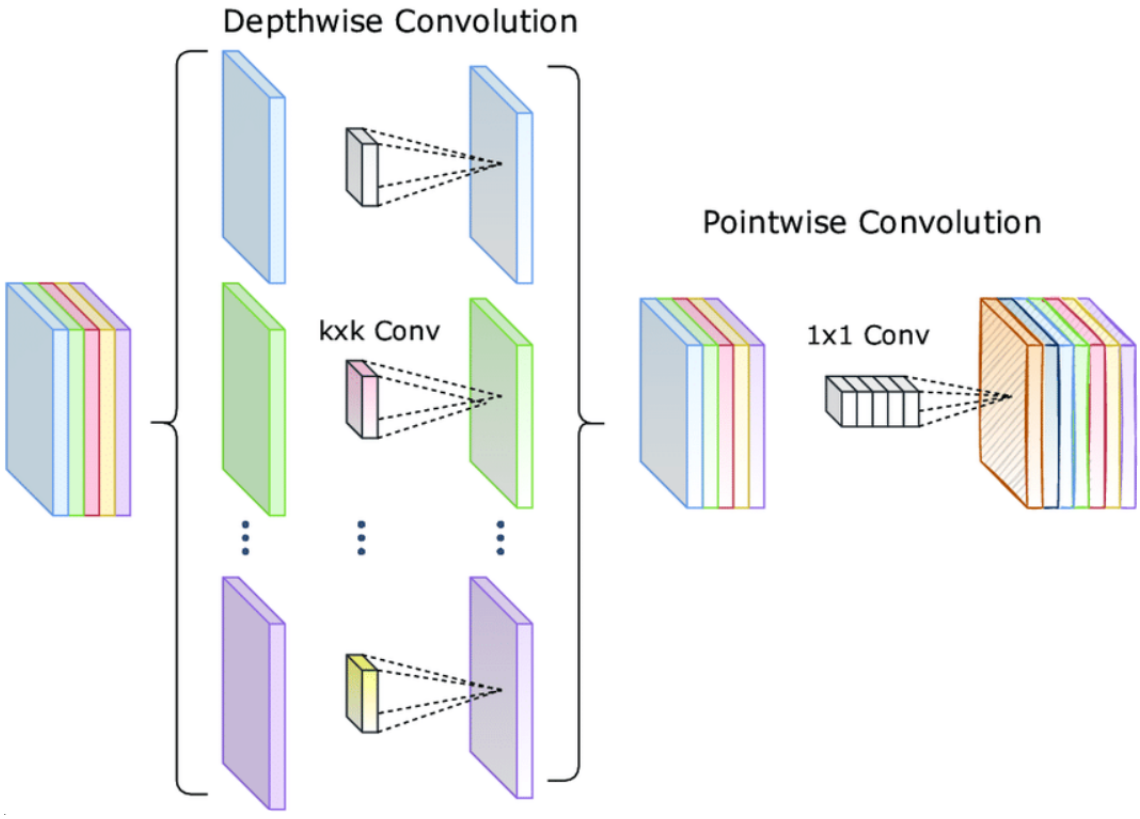


Figure 2.11: The depthwise convolution (left) applies a k×k filter independently to each input channel, while the pointwise convolution (right) uses a 1×1 filter.

2.5.3 Vision Transformers (ViT)

Introduction to Vision Transformers:

Vision Transformers (ViT) represent a shift from traditional convolutional architectures by adapting the Transformer model—originally designed for natural language processing—to the vision domain [6]. Instead of relying on convolution operations, ViTs process images by dividing them into fixed-size patches and treating these patches as tokens, much like words in a sentence (Figure 2.12).

How ViTs Work:

- **Patch Embedding:**

  The input image is split into a grid of patches (e.g., 16×16 pixels). Each patch is then flattened and linearly projected into a high-dimensional embedding space.

- **Positional Encoding:**

  Since Transformers do not inherently capture the spatial structure of the input, positional encodings are added to the patch embeddings to retain information about the position of each patch within the original image.

- **Self-Attention Mechanism:**

  The core of the Transformer is the self-attention mechanism, which computes pairwise interactions between all patches. This allows the model to capture global context and long-range dependencies, something that CNNs achieve only through very deep architectures.

- **Transformer Encoder Layers:**

  A stack of Transformer encoder layers processes the patch embeddings. Each layer includes multi-head self-attention and feed-forward networks, allowing the model to learn complex relationships among patches.

- **Output and Prediction:**

  For classification, a special "class token" is appended to the sequence of patch embeddings, and its final state is used as the image representation. For object detection, ViTs are often integrated with additional modules or combined with

CNN-based components (as seen in models like DETR) to generate bounding box predictions and class labels.

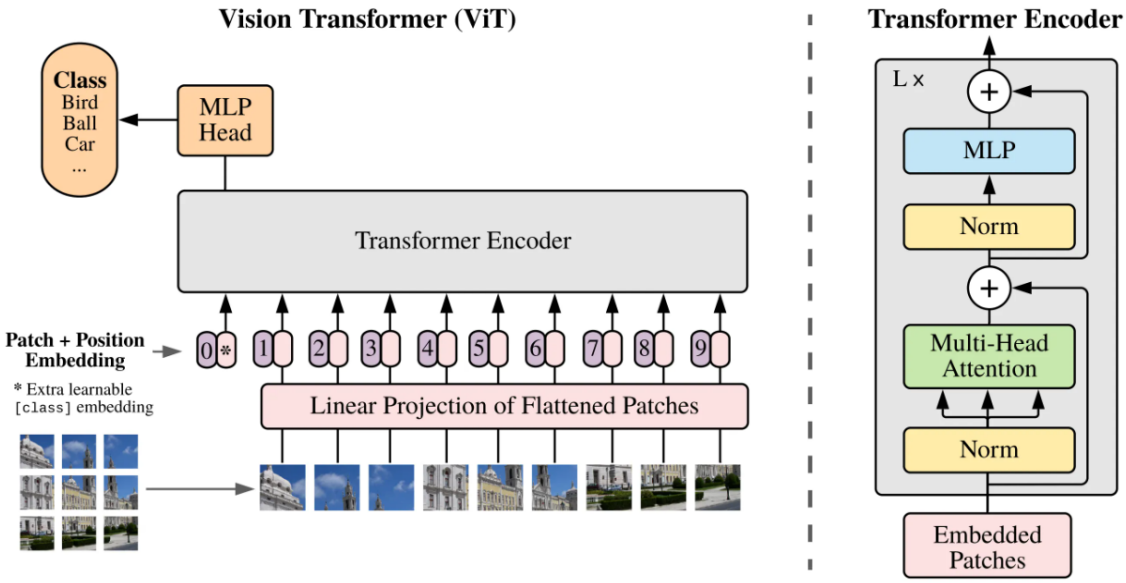


Figure 2.12: Overview of the Vision Transformer (ViT) architecture.

Advantages of Vision Transformers:

- **Global Context:**
  Self-attention enables ViTs to consider the entire image context when making predictions, which can improve the detection of objects that require global reasoning.

- **Scalability:**
  ViTs can leverage large-scale pre-training on massive datasets, leading to powerful representations that generalize well across tasks.

- **Flexibility:**
  The patch-based approach allows ViTs to be easily adapted to different image resolutions and sizes.

Challenges and Limitations:

- **Data Requirements:**
  ViTs typically require large amounts of training data to achieve competitive performance compared to CNNs.

- **Computational Cost:**
  Although scalable, the quadratic complexity of self-attention can be computationally demanding, especially for high-resolution images.

- **Lack of Built-In Inductive Biases:**
  Unlike CNNs, which have built-in biases for locality and translation invariance, ViTs rely entirely on learning these properties from data.

2.6 Foundation Models for Earth Observation

Modern remote sensing applications increasingly rely on deep learning to process and interpret large volumes of data collected by satellites. A major recent advancement is the development of foundation models—large-scale pre-trained models that synthesize and distill rich information from vast datasets. These models can then be fine-tuned for a wide range of downstream tasks, from object detection to change detection and beyond. In the context of Earth Observation (EO), foundation models not only reduce the need for task-specific training data but also enable robust transfer learning across diverse applications [19].

2.6.1 Overview of Foundation Models

Definition and Significance

Foundation models are large neural networks pre-trained on extensive, diverse datasets. Their purpose is to capture universal features and high-level representations that generalize well across many tasks. In remote sensing, such models learn from the spatial, temporal, and spectral information inherent in EO data, effectively "distilling" environmental knowledge into compact embeddings.

Key Characteristics

- **Universal Representations:** Foundation models learn to capture broad

and transferable features from data. For EO, this means encoding information about land cover, climate patterns, and surface structures into a high-dimensional space.

- **Self-Supervised Learning (SSL):** Many foundation models are trained using SSL techniques, such as masked autoencoding. This approach leverages unannotated data, making it possible to harness vast datasets without the need for extensive manual labeling.

- **Knowledge Transfer:** Once pre-trained, these models can be fine-tuned for various specific tasks. For instance, a foundation model trained on global satellite imagery can be adapted for object detection, classification, change detection, and other EO applications with minimal additional training.

- **Scalability and Efficiency:** By capturing a wide range of environmental variations, foundation models significantly reduce the need for task-specific models. They can serve as a backbone for many downstream applications, enabling more efficient and consistent performance across different datasets and conditions.

Importance in EO

In Earth Observation, the sheer volume and diversity of data—from different sensors and acquisition modes—make it challenging to develop dedicated models for every task. Foundation models overcome this challenge by providing a universal feature extractor that can be readily adapted to various applications, such as land cover classification, environmental monitoring, and disaster response. This paradigm shift not only streamlines model development but also enhances the overall robustness and generalizability of EO systems.

2.6.2 The Clay EO Model: Architecture and Operating Principles

Clay [7] is an open source foundation model for Earth Observation (EO). Foundation models trained on EO data can distill and synthesize vast amounts of environmental information, enabling them to generalize this knowledge across many downstream

applications such as object detection, classification, change detection, and more. Clay's key advantage is its ability to incorporate spatial, temporal, and spectral relationships from diverse satellite sources and produce high-level representations—or embeddings—of different regions on Earth's surface.

Purpose of Clay

- **Versatile EO Embeddings:** Clay ingests satellite imagery together with time and location metadata to produce embeddings, i.e., numeric representations capturing critical geospatial knowledge of an area at a particular time.
- **Self-Supervised Learning (SSL):** Clay is trained using a Masked Autoencoder (MAE) approach, allowing it to learn from unannotated data at a global scale.
- **Foundation Model for Nature & Climate:** By distilling knowledge of Earth's varied landscapes, Clay aims to make it easier to create applications that address climate change, biodiversity monitoring, and environmental management.

Architecture Overview

Clay uses a Vision Transformer (ViT) adapted for Earth Observation data. Some distinctive modules include:

1. **Dynamic Embedding Block:**
    - Generates patches from multi-band inputs, accounting for their specific wavelengths.
    - Handles data from various satellite systems (Sentinel-1, Sentinel-2, Landsat, etc.) with different band counts and resolutions.
2. **Position Encoding:**
    - Incorporates location (latitude, longitude), time (week/hour), and the Ground Sampling Distance (GSD).

- Scales positional embeddings by GSD to unify different spatial resolutions.

3. **Masked Autoencoder (MAE):**

   - A large ViT-based autoencoder that reconstructs masked patches, focusing on capturing fine geospatial and spectral details.
   - The MAE reconstruction loss forms about 95% of the total training loss.

4. **Teacher Model (DINOv2):**

   - Provides a representation loss (5% of total), helping the model learn better high-level embeddings in addition to the MAE objective.

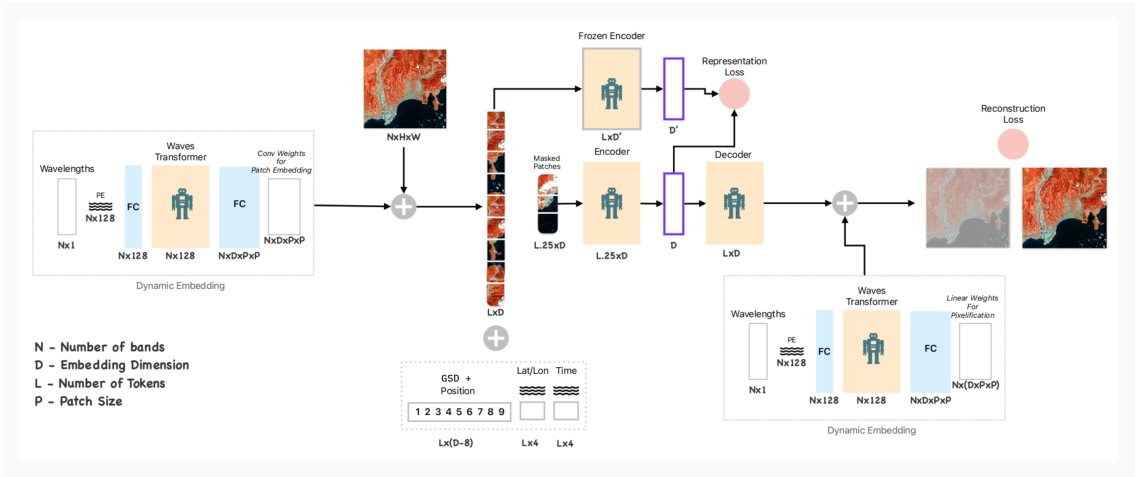


Figure 2.13: Architecture Overview of Clay Foundation Model.

Model Usage

Clay can be leveraged in three main ways:

1. **Generate Embeddings for Semantic Tasks:**

   - You can feed images (plus lat/lon/time, wave info) into the pretrained Clay model to obtain embeddings.
   - These embeddings support tasks like finding specific features (e.g., surface mines, deforestation, aquaculture sites) or analyzing changes over time.

2. **Fine-Tune for Downstream Tasks:**

- **Classification:** Distinguish land cover types, crop species, etc.
- **Regression:** Predict numeric values like above-ground biomass or vegetation indices.
- **Change Detection:** Identify temporal changes such as floods, wildfires, or urban growth.
- Fine-tuning can be done by training a smaller model on top of Clay's embeddings or by unfreezing Clay's own weights for even better specialization.

3. **Use Clay as a Backbone:**
   - Replace typical CNN backbones (e.g., ResNet) with Clay's pretrained transformer to handle spatiotemporal data more effectively.

Training Setup for Clay

- **Data Volume:** 70 million globally distributed training chips (images) of size 224×224, total size ∼33.8 TB.
- **Diverse Sensors:** Sentinel-1 and 2, Landsat 8/9, NAIP, LINZ, MODIS.
- **Global Coverage:** Emphasis on land surfaces, with sampling informed by land cover classes (ESA WorldCover), including wetlands, built-up areas, croplands, forests, etc.
- **Hardware:** 4 AWS EC2 p5.48x instances with 8 NVIDIA H100 GPUs each.
- **Training:** 10 epochs, each taking ∼50 hours to train.
- **Loss Components:**
  - MAE reconstruction: 95%
  - Representation (DINOv2 teacher): 5%

Chapter 3

Environment and Hardware Setup

3.1 Computational Environment and Libraries

The computational framework for this project was established using Python as the primary programming language. To manage dependencies and maintain reproducibility, virtual environments were created with Conda, ensuring that the project's libraries did not conflict and could be easily updated. Additional software were used:

- **GDAL:** An essential open-source library for geospatial data processing. GDAL was employed to read and manipulate GeoTIFF products, extract metadata (using commands such as `gdalinfo`).

- **QGIS:** [18] This open-source Geographic Information System was utilized for visual inspection and validation of georeferenced imagery. QGIS enabled interactive exploration of SAR products, verifying spatial accuracy and aiding in the identification of artifacts or anomalies in the data.

3.2 Hardware Specifications

Initial experiments were conducted on a workstation equipped with two NVIDIA RTX 3070 GPUs (each with 8GB of VRAM). This setup provided a robust environment for preliminary model development and testing. However, as the project progressed and the complexity of the models increased, it became necessary to leverage higher-performance computing resources. Subsequently, the project transitioned

to using ICEYE's dedicated server, `mlserver01`, which is outfitted with an NVIDIA Quadro RTX 8000 GPU featuring 48GB of VRAM. This upgrade was crucial for handling larger datasets and for training more advanced deep learning models, such as Clay, enabling more efficient experimentation and faster convergence.

The operating system used for these computations was a Linux-based distribution, which is standard in many scientific computing environments due to its stability and performance.

Chapter 4

Dataset and Preprocessing

This chapter describes the dataset used in this thesis and outlines the preprocessing steps applied to make the data suitable for training deep learning models. The dataset comprises ultra-high-resolution SAR images collected from ICEYE satellites, with the specific task of detecting airplanes across multiple acquisitions. Detailed procedures for data acquisition, labeling, and preprocessing are provided to ensure reproducibility and robustness of the experimental results.

4.1 ICEYE and Data Collection

ICEYE is a leading provider of SAR (Synthetic Aperture Radar) data, known for its capability to capture high-resolution images under all weather conditions and regardless of daylight. The ICEYE constellation employs advanced SAR sensors that offer diverse imaging modes, such as *Dwell*, *Dwell Fine*, and *Spot Fine*, making it possible to acquire detailed images of the Earth's surface. In this project, the ICEYE proprietary software was used to schedule image acquisitions by specifying precise geographic coordinates (latitude and longitude) and constraining the acquisition period to a short time window (typically 1–2 days). Additionally, the system allowed for the selection of specific satellites that support the required imaging formats, ensuring the collection of consistent and high-quality SAR images.

Acquisition Protocol

The acquisition process involved:

- **Geographic Specification:** Inputting exact coordinates for the target areas

to capture images where airplanes are likely to be present.

- **Temporal Constraints:** Limiting the acquisition period to ensure that images from multiple passes are temporally close. This aids in validating and cross-referencing data to improve the reliability of ground truth.

- **Sensor Selection:** Choosing satellites that support the desired imaging formats and acquisition mode (e.g., *Dwell*, *Dwell Fine*, *Spot Fine*), ensuring compatibility with subsequent preprocessing methods.

These measures helped in compiling a dataset comprising 2019 airplanes across 37 distinct acquisitions.

4.2 Data Labeling and Ground Truth Generation

4.2.1 Annotation Challenges

Labeling SAR images presents unique challenges that require a deep understanding of the underlying radar physics and imaging artifacts. Unlike optical images, SAR images are affected by phenomena that can significantly complicate the annotation process. Below are the primary challenges encountered during the annotation of SAR images for airplane detection:

- **Double Bouncing Artifacts:**
  SAR images often exhibit a "double bounce" effect. This occurs when an airplane is oriented such that its fuselage is perpendicular to the incoming radar wave. As a result, the radar signal may reflect off the ground and then off the airplane, or vice versa, causing the fuselage to appear twice in the image. This duplication makes it difficult to discern whether the bright signature corresponds to a single airplane or to another element entirely (Figure 4.1a).

- **Displacement Due to Elevation Angle:**
  Airplanes are typically large and tall relative to the surrounding terrain. In SAR imaging, the side-looking geometry causes significant displacement between an airplane and its shadow. The degree of displacement is heavily

influenced by the satellite's elevation angle. This misalignment between the physical position of the airplane and its shadow complicates the accurate placement of bounding boxes (Figure 4.1b).

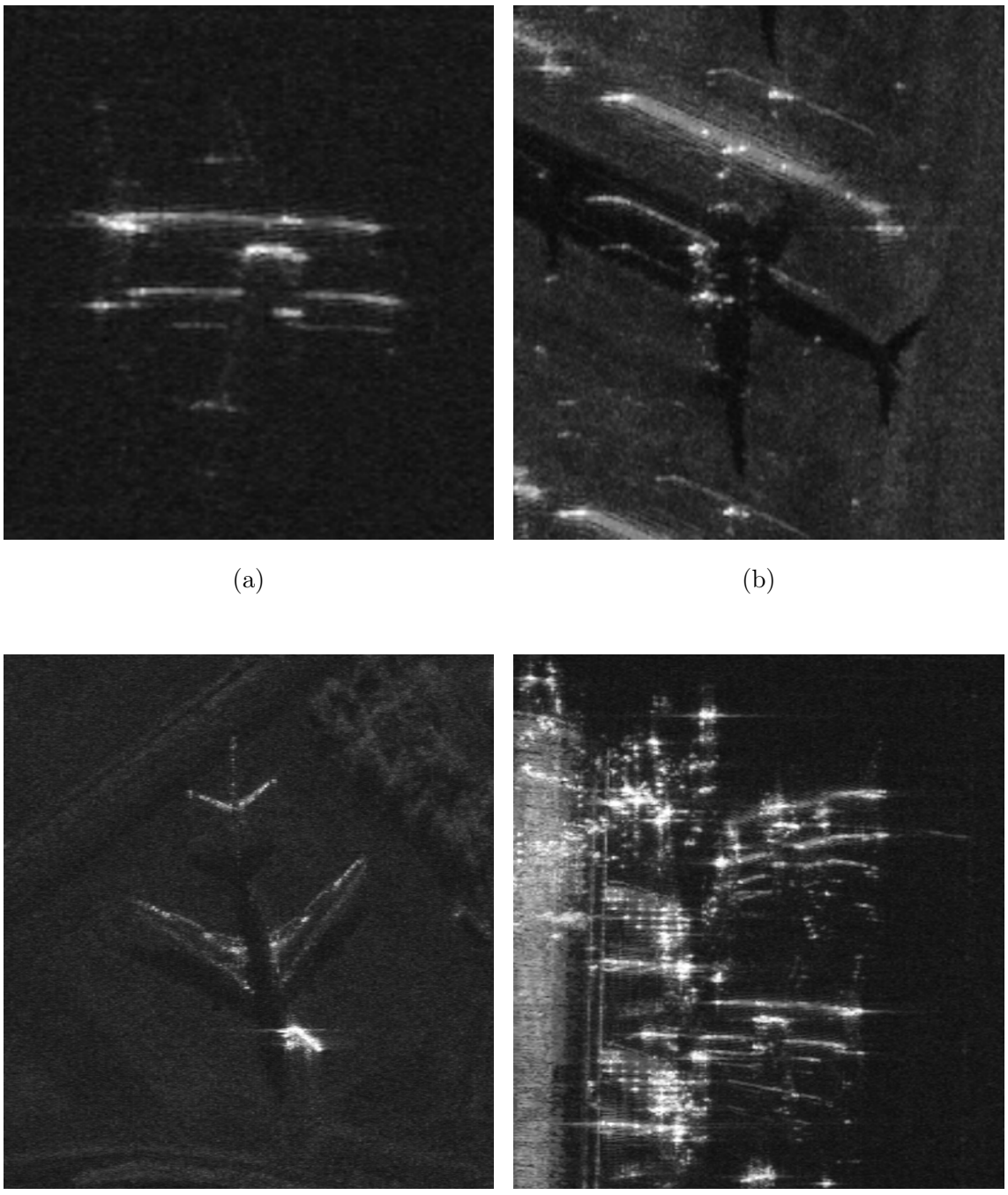
- **Identification of High-Altitude Features:**
  Some parts of the airplane, such as the tail, are particularly challenging to label. Typically, the tail is much higher than the main body of the airplane. In SAR images, this height difference causes the tail to appear significantly more displaced from the main fuselage compared to other parts of the airplane. Moreover, in many cases, the tail may not reflect the radar signal well and thus appears as only a small, bright spot. This small and significantly displaced reflection makes it difficult to include the tail accurately within the bounding box, risking its exclusion from the annotation (Figure 4.1c).

- **Complex Scene Composition:**
  While isolated airplanes can be annotated with relative ease, scenes that contain multiple complex structures—such as boarding stairs or airport gates—introduce additional ambiguity. When airplanes are adjacent to or partially obscured by other objects, distinguishing the precise boundaries of each airplane becomes much more difficult. The presence of multiple interfering elements requires annotators to exercise extra caution to avoid mislabeling and ensure that bounding boxes accurately capture the target object (Figure 4.1d).

4.2.2 Ground Truth Generation

Establishing accurate ground truth in SAR imagery is a critical yet challenging task. In my work, several strategies were employed to ensure that each airplane is reliably identified and its corresponding bounding box is correctly positioned. Key aspects of our approach are described below:

- **Utilizing Shadows for Identification**



(a) (b)

(c) (d)

Figure 4.1: (a) The image depict an airplane characterized by the double bouncing phenomenon. (b) The image depict an airplane shifted from his shadow. (c) The image depict an airplane with an high tail artifact. (d) A complex scene where the airplane is close to the Airport's gate.

The shadow cast by an airplane often serves as a crucial cue for confirming its presence. In many instances, the shadow enhances the contrast between the airplane and its surroundings, making it easier to delineate the object. However, in terrains with not well diffuse reflectivity, the shadow may blend into the uniformly dark background, thereby reducing its usefulness as a distinguishing feature (Figure 4.2a) (Figure 4.2b).

- **Multi-Acquisition Cross-Validation**

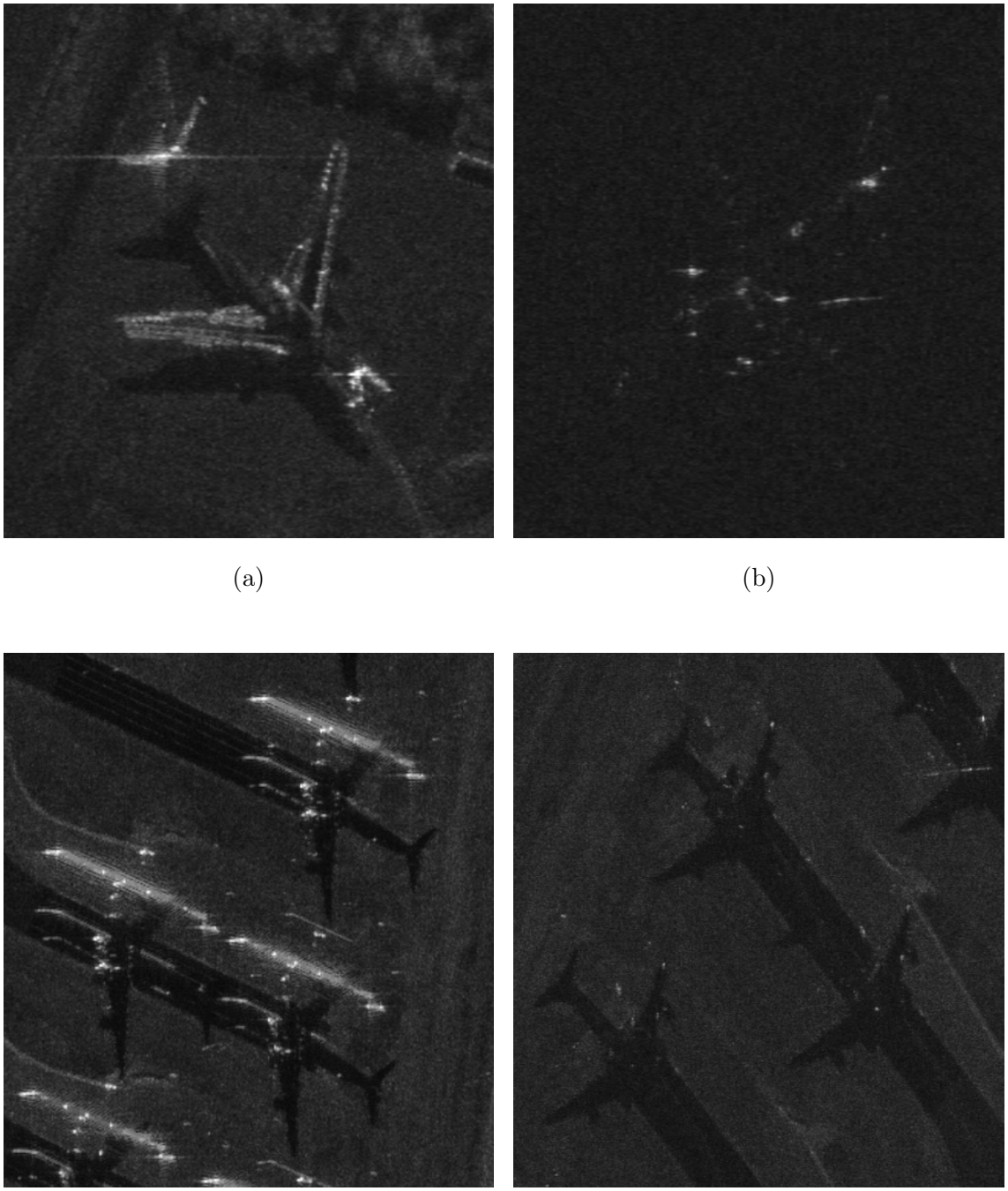
  To overcome the variability in radar reflectivity caused by different viewing angles, multiple acquisitions of the same area were obtained within a short time window (typically a few hours). These acquisitions, captured at varying angles, allowed for a more robust determination of the airplane's true position. When an airplane did not reflect strongly in one acquisition, another acquisition might capture a strong reflection, thereby enabling the correct placement of a bounding box (Figure 4.2c) (Figure 4.2d).

- **Verification with Ancillary Data**

  To further enhance the reliability of ground truth annotations, Google Earth Pro was employed as an auxiliary verification tool. By cross-referencing the SAR imagery with high-resolution optical imagery from Google Earth Pro, I was able to distinguish between actual airplanes and other objects (such as buildings or vehicles) that might have similar radar signatures. This step was essential to avoid erroneous labeling and ensure that only genuine airplanes were annotated (Figure 4.3).

4.3 Data Preprocessing

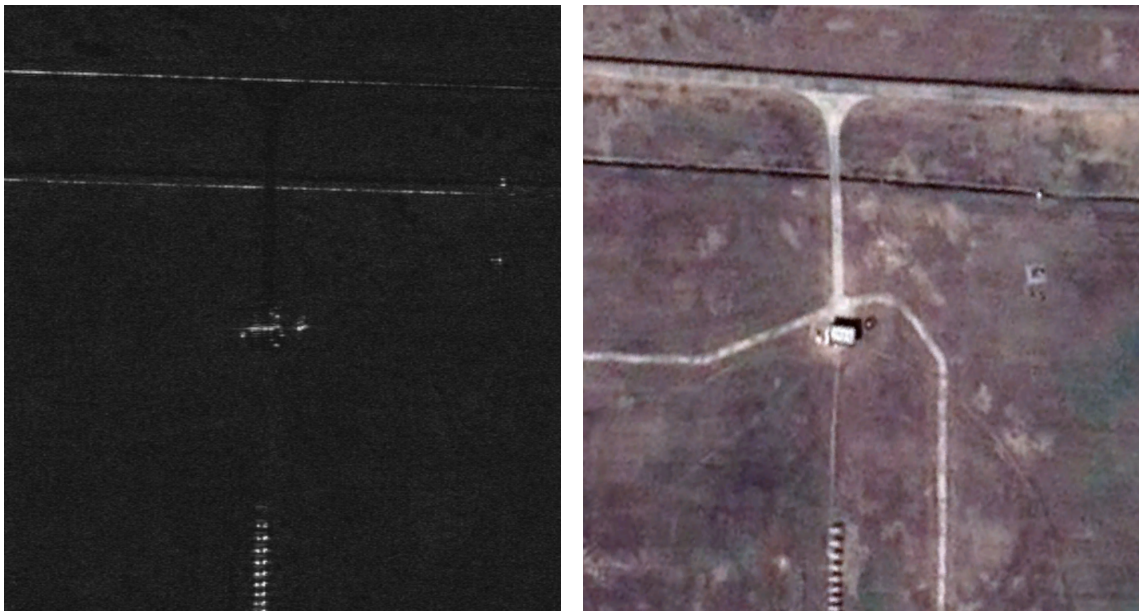
To prepare the ICEYE SAR data for our experiments, several preprocessing steps were required to transform the original ultra-high-resolution images into formats that are manageable by deep learning models. The ICEYE data is delivered in two



(a) (b)

(c) (d)

Figure 4.2: (a) (b) An example where the airplane's shadow is clearly visible versus one where it is obscured by low-reflectivity terrain. (c) (d) Examples showing how images captured at different angles complement each other, ensuring that even if one image has weak reflections, another provides sufficient detail to confirm the presence and position of the airplane.



(a) (b)

Figure 4.3: Google Earth imagery helped disambiguate an object that resembled an airplane in the SAR data.

proprietary formats: Ground Range Detected (GRD) and Colorized Sub-aperture Image (CSI). Each format necessitates specific processing to address memory constraints, calibration, and standardization of resolution.

4.3.1 Format Conversion and Calibration

A critical step in our preprocessing pipeline is converting ICEYE GRD (Ground Range Detected) images into a standardized format suitable for both visualization and deep learning (e.g., PNG). This conversion process is designed to optimize the raw radar data by applying a logarithmic transformation, statistical clipping, and 8-bit scaling. The process is implemented through a series of functions, as illustrated below:

- **Logarithmic Transformation:**

  The raw GRD data, which typically has a very high dynamic range, is first transformed into a decibel scale using the formula:

  $$\text{grd\_log} = 10 \times \log_{10}(\text{grd} + \epsilon)$$

  Here, ϵ is a small constant (e.g., 1×10^{-6}) to avoid taking the logarithm of

zero. This log transformation compresses the dynamic range, making subtle variations in backscatter more visible.

- **Clipping Based on Statistical Distribution:**

  The function `clip_to_distribution` calculates the mean (μ) and standard deviation (σ) of the log-transformed image, ignoring any NaN values. It then defines clipping thresholds as:

$$\text{min\_val} = \lfloor \mu - 4\sigma \rfloor \quad \text{and} \quad \text{max\_val} = \lfloor \mu + 4\sigma \rfloor$$

  This step helps to mitigate the impact of extreme outliers by confining the pixel values to a range where most of the data lies.

- **Conversion to 8-bit Format:**

  The function `convert_float_to_8` takes the clipped image and rescales it linearly to an 8-bit format. The scaling factor is computed as:

$$\text{scale} = \frac{255}{\text{max\_val} - \text{min\_val}}$$

  The image is then clipped to the $[\text{min\_val}, \text{max\_val}]$ range and transformed as follows:

$$\text{image\_8bit} = \left(\frac{\text{image} - \text{min\_val}}{\text{max\_val} - \text{min\_val}} \times 255 \right)$$

  This conversion ensures that the image values lie within the $[0, 255]$ interval, making them suitable for visualization and subsequent processing.

See Appendix (List C.2) for the complete code of the GRD conversion pipeline.

4.3.2 Image Tiling and Resolution Adjustment

Ultra-high-resolution ICEYE images, typically sized at more than 30000×30000 pixels, are too large to be processed directly by deep learning architectures without overwhelming memory resources. To make these images manageable for both convolutional and transformer-based models, we perform two key operations: resolution adjustment (resizing) and tiling with overlap.

- **Resolution Adjustment:**

In the first step, the original images are resized to several target resolutions. This standardization not only reduces the overall data volume but also ensures consistent spatial resolution across different acquisitions. By generating datasets at multiple resolutions (e.g., 4080×4080, 8160×8160, and 16320×16320 pixels), we can analyze how image scale affects detection performance while reducing the computational burden during training.

- **Image Tiling with Overlap:**

  After resizing, the images are segmented into smaller tiles to further alleviate memory constraints. Each tile is extracted with a predefined size and, importantly, with overlapping regions between adjacent tiles. The overlap is crucial: it ensures that objects appearing at the boundaries of one tile are also captured in neighboring tiles, preventing potential loss of critical information. Moreover, tiles without any label weren't included in the final dataset.

Chapter 5

YOLO for Object Detection in SAR Imagery

5.1 Introduction

5.1.1 Motivation for Applying YOLO to SAR Data

During my internship project, YOLO [17] was chosen as the initial model for object detection on SAR imagery primarily due to its ease of use, lightweight nature, and low computational and memory requirements. This choice enabled rapid prototyping and extensive hypothesis testing, even under the limited resource constraints initially provided by ICEYE. YOLO's straightforward architecture allowed for quick iterations and experiments, making it particularly suitable for exploring diverse aspects of SAR data. Moreover, YOLO's pretraining—originally on large-scale optical datasets—offered a useful baseline for evaluating its transferability to SAR imagery. This facilitated direct comparisons between different SAR formats, such as GRD (grayscale) and CSI (colorized), and helped assess whether the color information in CSI images enhanced detection performance over traditional GRD images. Additionally, numerous experiments were conducted to determine the optimal patch size for detection.

5.1.2 Overview of Custom Adaptations for SAR Object Detection

To effectively apply YOLO to SAR data, several custom adaptations were incorporated into the standard YOLO pipeline. These modifications addressed the unique challenges posed by SAR imagery, including speckle noise, geometric distortions, and the distinct radiometric characteristics of SAR data. Key adaptations include:

- **Image Format Evaluation:**
  Experiments were performed on both GRD and CSI formats to understand the impact of grayscale versus color information. This comparison was critical in determining whether the pretraining on optical datasets could be effectively transferred to SAR imagery and whether the additional spectral information in CSI images could improve detection accuracy.

- **Patch Extraction and Spatial Context:**
  Given that the original SAR acquisitions cover a fixed area of 5 km $\times$ 5 km at varying resolutions, the images were segmented into patches to standardize the input size for the model. Overlapping patches were used to ensure that objects located at the boundaries were not missed, providing a consistent spatial context for detection.

- **Pretraining Evaluation:**
  The pretraining of YOLO, primarily on optical data, was scrutinized to gauge its effectiveness when applied to SAR images. This evaluation helped identify whether the features extracted by the backbone were sufficient for detecting small objects such as airplanes.

Figure 5.1: Schematic diagram of the adapted YOLO pipeline for SAR imagery. The diagram illustrates the workflow from image tiling and patch extraction (ensuring each patch covers a fixed ground area of 5 km $\times$ 5 km), through the evaluation of GRD (grayscale) and CSI (colorized) image formats, to the final object detection output.

This adapted pipeline provided a robust framework to investigate YOLO's performance in the challenging domain of SAR imagery, ultimately laying the groundwork

for subsequent enhancements and alternative backbone evaluations.

5.2 Experimental Setup and Evaluations

In this section, we describe the experimental framework used to evaluate the performance of YOLO for object detection on SAR imagery. Our experiments were designed to explore several key aspects of the detection pipeline, including model complexity, the influence of image resolution and spatial context, the role of color information in SAR images, and the impact of various data augmentation strategies. Each experiment was conducted while maintaining a consistent preprocessing pipeline and using the same underlying dataset, which consists of 2019 airplane instances across 37 acquisitions. The dataset was divided in train (80%) and validation (20%). Detailed results, along with corresponding figures, are provided to illustrate the performance trends and to support our analysis.

5.2.1 Model Complexity Analysis

Hypothesis:

We hypothesized that increasing the model complexity within the YOLOv8 family would enable the network to capture finer details in SAR imagery, potentially leading to higher detection accuracy. However, this benefit might be counterbalanced by increased training instability and a heightened risk of overfitting—particularly when working with a limited dataset.

Methodology:

To evaluate this, we conducted experiments using the complete set of YOLOv8 variants: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. The experiments utilized grayscale SAR images at 4096×4096 resolution. Each image was partitioned into patches of 640×640 pixels with a 128-pixel overlap, ensuring that objects located at the boundaries were not missed. All variants were trained under identical conditions for 500 epochs, with consistent preprocessing, data augmentation, and hyperparameter settings.

Results:

As illustrated in (Figure 5.2), the larger models (YOLOv8l and YOLOv8x) demon-

strated higher accuracy, likely due to their increased capacity for feature extraction. However, these models also showed signs of early training instability, which we attribute to overfitting given the limited training data. In contrast, the smaller models (YOLOv8n and YOLOv8s) exhibited more stable convergence and lower computational overhead, though at the cost of slightly reduced accuracy. The mid-level YOLOv8m variant struck a balance between complexity and stability, performing well overall.

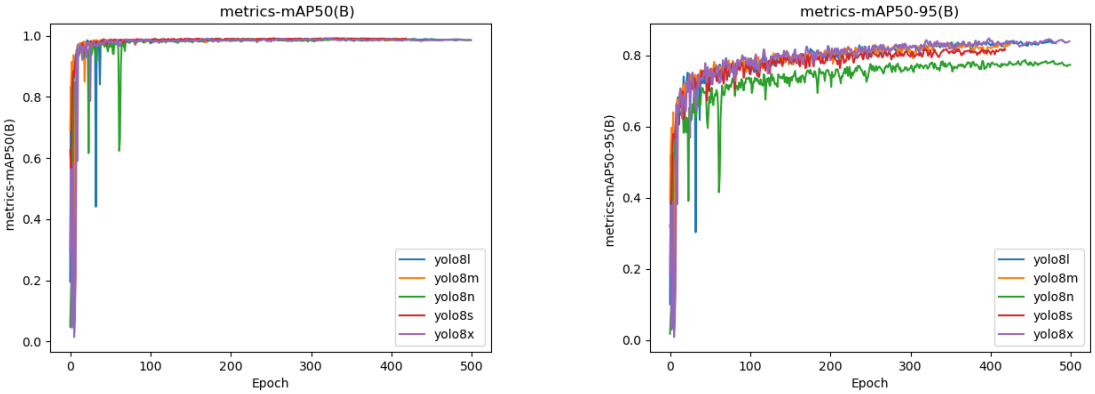


Figure 5.1: Yolo Complexity Color results.

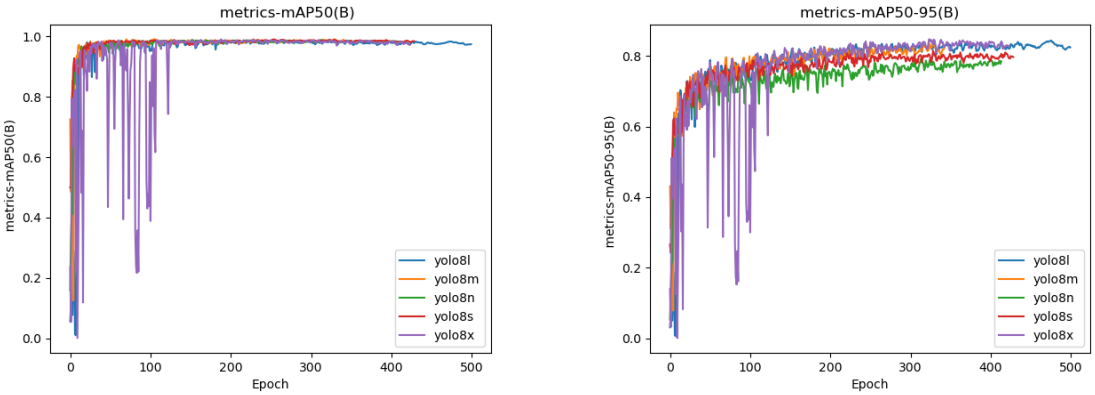


Figure 5.2: Yolo Complexity Grayscale results.

5.2.2 Impact of Spatial Context

Hypothesis:

We hypothesized that increasing the spatial context within the input patches would improve the detection performance for airplanes in SAR imagery. In particular, including a broader environmental view (e.g., airport infrastructure such as runways, terminals, and taxiways) can serve as contextual cues that help the model distinguish true airplane detections from false positives. Conversely, a limited spatial context might hinder the network's ability to discern whether an object truly belongs to an airport scene.

Methodology:

To investigate this, we conducted experiments on images with a fixed resolution of 4096×4096 pixels, extracted into patches of varying sizes: 640×640, 1024×1024, 2048×2048, and 4096×4096 pixels. Overlapping regions were maintained between patches to ensure that objects located at the boundaries were not missed. This experimental design allowed us to isolate the effect of spatial context—i.e., the amount of surrounding area included in each patch—on detection accuracy.

Results:

Our results, as shown in (Figure 5.3), indicate that patches with an intermediate spatial context (e.g., 1024×1024 or 2048×2048 pixels) yield a better balance between local detail and contextual information. Patches that were too small lacked the necessary environmental cues, leading to increased false positives, whereas extremely large patches, despite containing detailed information, resulted in a reduced number of training samples, which slowed convergence and occasionally diluted object-specific features. These findings support the notion that a moderate amount of surrounding context is beneficial for correctly localizing and classifying airplanes in SAR imagery.

5.2.3 Impact of Different Resolutions

Hypothesis:

We hypothesized that higher resolution images would provide more detailed features and finer object details, thereby improving the detection performance for small objects such as airplanes. However, this benefit might be counterbalanced by increased computational costs.

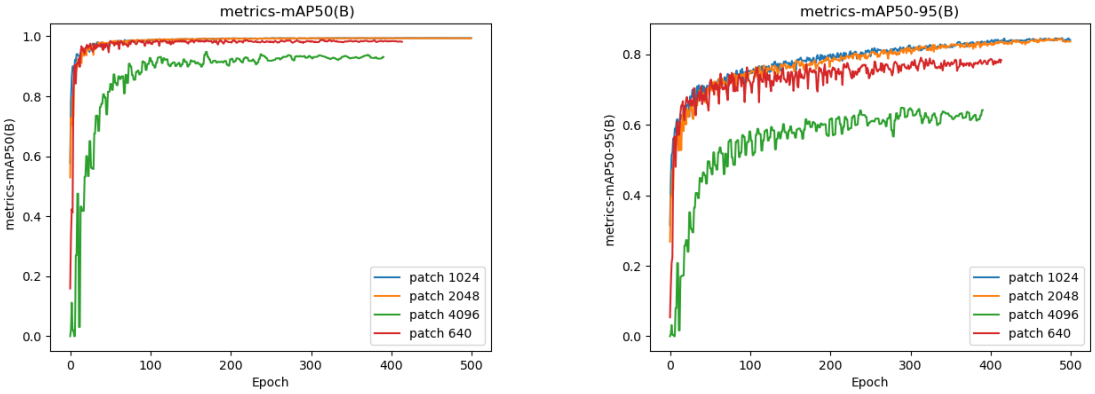


Figure 5.3: Impact of Spatial Context results.

Methodology:

Starting from the original SAR acquisitions, which cover a fixed ground area of 5 km × 5 km, we generated multiple datasets by scaling the images to different resolution levels. For each resolution level, the images were resized and then divided into patches while maintaining a constant physical context. For example, for images originally sized at 4096×4096 pixels, we extracted patches of 1024×1024 pixels; similarly, for images scaled to 8192×8192 pixels, patches of 2048×2048 pixels were used. Identical preprocessing, augmentation, and training procedures were applied across all resolution datasets to ensure a controlled comparison.

Results:

As shown in (Figure 5.4), the experiments demonstrated that higher resolution images indeed provided improved detection accuracy due to the increased level of detail available for feature extraction. However, these benefits were accompanied by higher computational and memory requirements. The results clearly indicate that while a higher resolution can enhance the detection performance, an optimal trade-off must be achieved to balance accuracy and resource consumption.

5.2.4 Evaluation of Color versus Grayscale SAR Images

Hypothesis:

We hypothesized that incorporating color information from the CSI format might enhance the object detection performance compared to traditional grayscale images

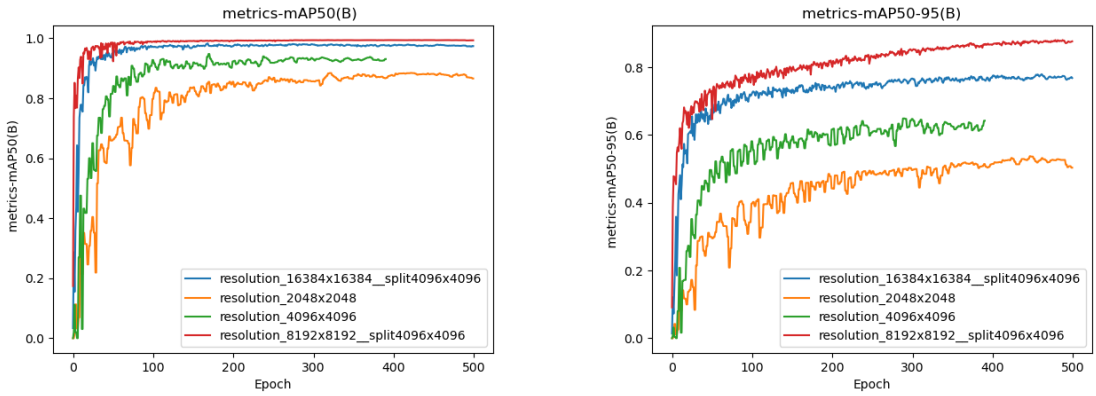


Figure 5.4: Impact of Different Resolutions results.

(GRD). The expectation was that the additional spectral cues in CSI could help the network better differentiate features, potentially leading to improved classification accuracy.

Methodology:

To evaluate this, we prepared two datasets from the same SAR acquisitions: one using the GRD (grayscale) format and another using the CSI (Color Sub-Aperture Image) format. Both datasets underwent identical preprocessing, which included scaling to a consistent resolution, patch extraction while maintaining a fixed physical ground area, and applying the same augmentation strategies. The YOLOv8 model was then trained on both datasets under the same hyperparameter settings. This controlled setup allowed us to directly compare the impact of color information on detection metrics.

Results:

As shown in (Figure 5.5), the experimental outcomes revealed that the performance difference between the CSI and GRD datasets was marginal. Although the CSI images provided extra spectral information, this did not translate into a significant improvement in detection accuracy for airplanes. It appears that the color cues in CSI do not offer substantial advantages over grayscale representations for this specific task. It is also important to consider that the meaning of color in SAR images differs significantly from the color semantics learned during pretraining on the COCO dataset.

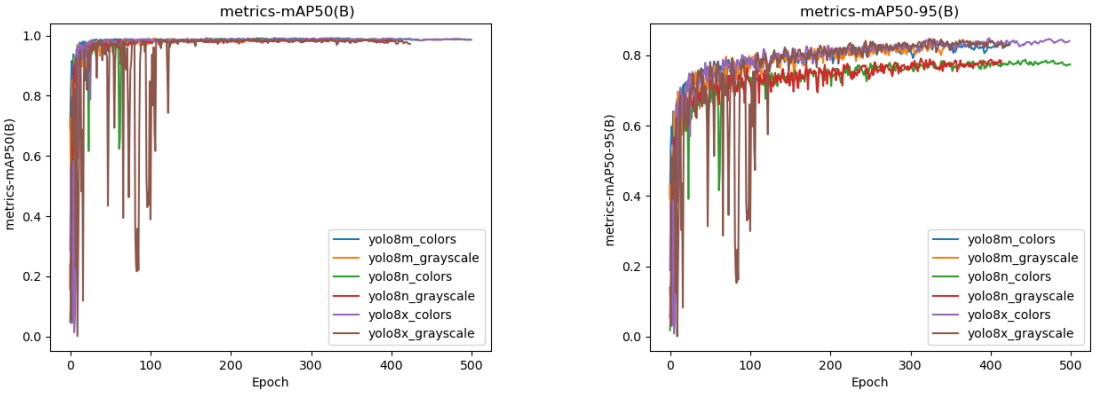


Figure 5.5: Color versus Grayscale results.

5.2.5 Data Augmentation Strategies

Hypothesis:

We hypothesized that applying a standard data augmentation pipeline to SAR images might improve model generalization and robustness compared to training without any augmentation. However, given the unique characteristics of SAR data, there is also a possibility that standard augmentation methods may be insufficient, and that specialized augmentation techniques tailored to SAR imagery might be required.

Methodology:

For this experiment, we utilized grayscale SAR images in the GRD format from the same dataset. Two experimental setups were compared: one using a standard data augmentation pipeline and one with no augmentation at all. Identical preprocessing, training, and hyperparameter configurations were maintained across both setups to ensure a fair comparison.

Results:

As illustrated in (Figure 5.6), the model trained with standard data augmentation achieved a modest improvement in mean average precision (mAP) over the model trained without any augmentation. The augmented model also demonstrated enhanced training stability and a reduced tendency to overfit. Nonetheless, the performance gains suggest that while standard augmentation offers benefits for SAR data, there may be further room for improvement through augmentation strategies specifically designed to account for SAR-specific noise and geometric distortions.

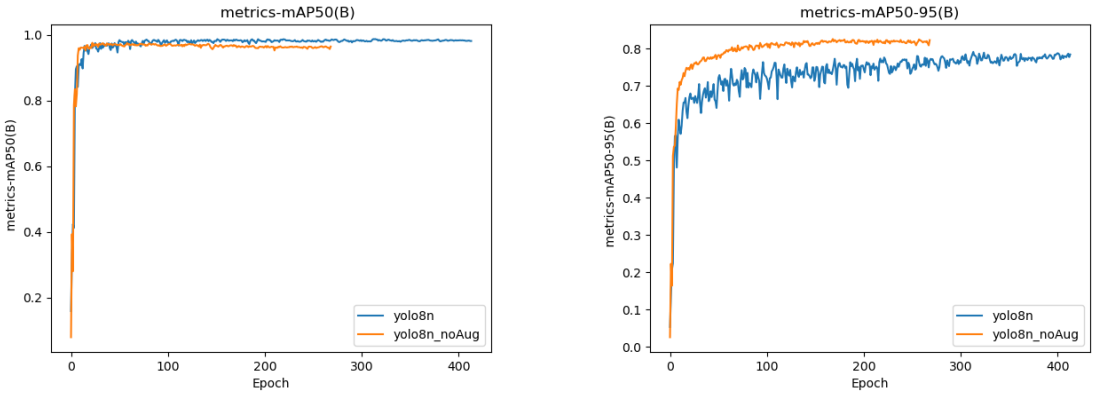


Figure 5.6: Data Augmentation Strategies results.

Chapter 6

Clay Enhancements for Object Detection

In this chapter, we describe the modifications made to extend the Clay foundation model for object detection tasks. Although Clay was originally designed to generate high-level embeddings for Earth observation, our project required it to perform precise bounding box regression and classification. To achieve this, we appended a custom detection head and implemented several additional enhancements. These modifications include the design of a convolutional architecture for the detection head, various head configuration options, fine-tuning strategies, specialized loss functions, and target assignment techniques. Figures referenced throughout this chapter illustrate the overall architecture and key design choices.

6.1 Adding an Object Detection Head

To repurpose the Clay model for object detection, we integrated a dedicated detection head that processes the 2D feature maps (reshaped from [B, L, D] to [B, D, H, W]) generated by Clay. This head is responsible for predicting bounding box coordinates and class scores for each object in the scene. The design is inspired by modern single-stage detectors like YOLOv8 and follows a grid-based approach.

6.1.1 Convolutional Architecture for the Detection Head

The detection head is built as a series of convolutional blocks that refine the feature maps output by the backbone. Each block follows a consistent architecture pattern, referred to as the CBS block (Figure 6.1), which consists of:

- **Convolutional Layer:**
  A 2D convolution is applied with a kernel size of 3×3, a stride of 1, and padding of 1. This setting preserves the spatial resolution of the input feature map while enabling the extraction of local spatial features.

- **Batch Normalization:**
  Immediately following the convolution, batch normalization is applied. This step stabilizes and accelerates the training process by normalizing the activations, reducing internal covariate shift, and allowing for higher learning rates.

- **SiLU Activation:**
  The normalized output is then passed through the SiLU (Sigmoid Linear Unit) activation function, also known as Swish. The SiLU function introduces nonlinearity and has been empirically shown to outperform other activation functions such as ReLU in various tasks.

After each CBS block, a pointwise convolution (1×1 convolution) is employed to reduce the channel dimensionality to the desired number of features.

Mathematically, each convolutional operation can be represented as:

$$Y_{i,j} = \text{SiLU}\left(\text{BN}\left(\sum_{m,n} W_{m,n} \cdot X_{i+m,j+n} + b\right)\right)$$

where X is the input feature map, W and b are the weights and bias of the convolutional layer, and Y is the output feature map.

This convolutional architecture is crucial as it provides the detection head with the capacity to learn intricate spatial features from the pre-trained embeddings while preserving the global contextual information derived by the Transformer backbone.

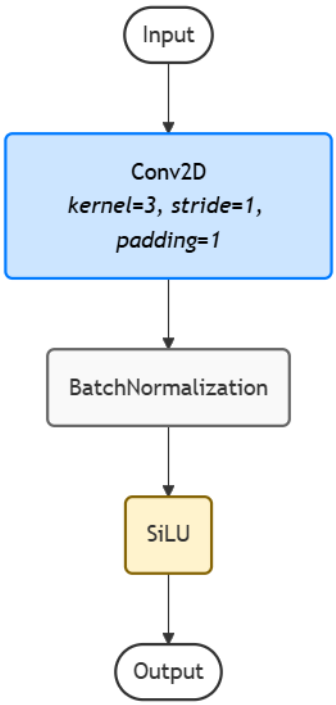


Figure 6.1: CSB Block.

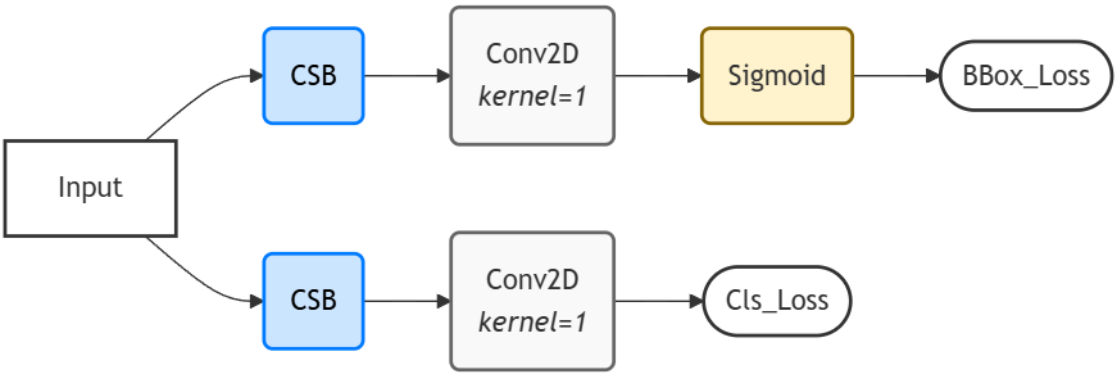


Figure 6.2: Decoupled Convolutional Head.

6.1.2 Head Configurations

In designing the detection head, several configuration strategies were explored to determine the most effective method for predicting bounding boxes and class scores. Two main configurations were compared:

1. **Center-Width-Height (CW-H) Configuration:**

   In this configuration, each predicted bounding box is represented by four parameters:

   - (x_c, y_c): Normalized center coordinates of the bounding box.
   - (w, h): Normalized width and height of the bounding box.

   A Sigmoid activation function is applied to ensure that all predicted values are constrained within the range $[0, 1]$. This approach aligns with common YOLO-like architectures and has been shown to be effective in localizing objects precisely.

2. **Corner Coordinate Configuration:**

   An alternative approach involved predicting the bounding box using the coordinates of the top-left and bottom-right corners, represented as (x_1, y_1, x_2, y_2). Although this method has its theoretical merits, in our experiments it did not result in significant improvements in detection accuracy compared to the CW-H approach. Based on extensive testing and evaluation against ground truth annotations, the CW-H configuration was ultimately adopted due to its robustness and consistency in our detection tasks.

In addition to the choice of configuration, a critical design decision was to apply separate convolutional layers for bounding box regression and classification. The regression branch concludes with a pointwise convolution that outputs num_box $\times$ 5 values (for each bounding box: x_c, y_c, w, h, and objectness score), while the classification branch outputs num_box $\times$ num_classes scores. This separation allows the

model to specialize in spatial localization and semantic classification independently, thereby improving overall performance.

6.2 Fine-Tuning Strategy

After integrating the custom detection head with the Clay foundation model, the next step is to fine-tune the entire network for the object detection task. Fine-tuning involves adapting the pretrained weights (both in the backbone and the detection head) on a task-specific dataset to improve performance on our target domain—in this case, detecting airplanes in high-resolution SAR imagery. This section describes the training procedure, hyperparameter configuration, and the methods we adopted to ensure robust learning while mitigating overfitting.

6.2.1 Training and Hyperparameter Configuration

For fine-tuning, we used the AdamW optimizer due to its robustness in handling sparse gradients and decoupled weight decay. Our hyperparameter setup was chosen to balance convergence speed with training stability. The key settings are as follows:

- **Learning Rate (α):** Set to 1×10^{-4}. This value was selected after empirical tuning to ensure gradual convergence without overshooting the optimal minima.
- **Weight Decay (λ):** Set to 1×10^{-4} to regularize the weights and reduce overfitting.
- **Betas:** The momentum parameters were set as $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to effectively control the exponential decay rates for the first and second moments of the gradients.
- **Non-Maximum Suppression (NMS) Thresholds:**
  - **Confidence Threshold:** 0.25 to filter out low-confidence detections.
  - **IoU Threshold:** 0.45 to remove redundant bounding boxes.

The training process was conducted over 500 epochs using our dataset, which comprises 37 acquisitions with 2019 airplane samples. Due to the high memory cost of the encoder, only low-resolution images of 4096×4096 pixels were utilized. These images were divided into 640×640 pixel patches with an overlap of 128 pixels between adjacent patches, facilitating efficient processing without compromising prediction quality. During training, we monitored loss curves, mean Average Precision (mAP), and precision–recall metrics on both the training and validation sets. These metrics allowed us to adjust hyperparameters dynamically and implement early stopping or learning rate scheduling (e.g., StepLR) when necessary to prevent overfitting.

6.3 Custom Loss Functions for Object Detection

To effectively train the object detection head, a composite loss function was implemented. This loss function integrates multiple components to address different aspects of object detection:

Objectness Loss

- **Purpose:**

  Evaluates the model's ability to correctly predict the presence of an object within a specific bounding box and grid cell.

- **Components:**

  - **Positive Samples (obj_loss):**

    Implementation: Utilizes Binary Cross Entropy (BCE) loss between the predicted objectness score ($\hat{o}$) and the target objectness (o) for boxes responsible for detecting objects (obj_mask).

  - **Negative Samples (noobj_loss):**

    Implementation: Applies Binary Cross Entropy (BCE) loss for boxes that do not correspond to any object (noobj_mask), encouraging the model to predict low objectness scores for these regions.

$$\text{obj\_loss} = \frac{1}{N_{\text{obj}}} \sum_{i \in \text{obj\_mask}} \Big[o_i \, \log\big(\sigma(\hat{o}_i)\big) \; + \; \big(1 - o_i\big) \, \log\big(1 - \sigma(\hat{o}_i)\big) \Big]$$

$$\text{noobj\_loss} = \frac{\lambda_{\text{noobj}}}{N_{\text{noobj}}} \sum_{i \in \text{noobj\_mask}} \Big[o_i \, \log\big(\sigma(\hat{o}_i)\big) \; + \; \big(1 - o_i\big) \, \log\big(1 - \sigma(\hat{o}_i)\big) \Big]$$

Coordinate Loss

- **Purpose:**

  Ensures accurate regression of the bounding box coordinates, specifically the center (x_c, y_c) and the dimensions (w, h) of the predicted boxes.

- **Components:**

  - **Complete Intersection over Union (CIoU) Loss:**

    It captures overlap, centroid distance, and aspect ratio between the predicted and ground truth boxes.

  - **Mean Squared Error (MSE) Loss for Width and Height:**

    It penalizes deviations in the width and height predictions.

$$\text{coord\_loss} = \lambda_{\text{coord}} \sum_{i \in \text{obj\_mask}} \big(1 - \text{CIoU}_i\big) \; + \; \frac{\lambda_w}{N_{\text{obj}}} \sum_{i \in \text{obj\_mask}} \Big[\big(w_t - w_i\big)^2 + \big(h_t - h_i\big)^2 \Big].$$

Classification Loss

- **Purpose:**

  Evaluates the model's ability to correctly classify the detected objects.

- **Components:**

  Utilizes Binary Cross Entropy with Logits (BCE with logits) between the predicted class scores ($\hat{c}$) and the target classes (c) for boxes responsible for object detection (obj_mask).

$$\text{cls\_loss} = - \sum_{i \in \text{obj\_mask}} \sum_{c=1}^{C} \Big[c_{i,c} \, \log\big(\sigma(\hat{c}_{i,c})\big) \; + \; \big(1 - c_{i,c}\big) \, \log\big(1 - \sigma(\hat{c}_{i,c})\big) \Big].$$

Total loss: Sum of the partial losses divided by the batch size.

$$\text{Total Loss} = \frac{1}{B} \big(\text{obj\_loss} + \text{noobj\_loss} + \text{coord\_loss} + \text{cls\_loss}\big).$$

6.4 Matching: Target Assignment Strategy

Accurate object detection relies on effectively associating each ground truth (GT) object with one of the predicted bounding boxes. In our framework, this association is performed by the `build_target` function, which assigns responsibility for each GT box to a specific prediction based on its location in the output grid and the Intersection over Union (IoU) metric. This matching process is critical because it directly influences the supervision signal during training (Code C.3).

The matching process follows these key steps:

1. **Initialization:**

   Several target tensors are initialized:

   - *Object Mask (`obj_mask`):* A boolean tensor indicating which predicted box is responsible for a GT object.
   - *No-object Mask (`noobj_mask`):* A boolean tensor set to True for boxes that do not correspond to any object.
   - *Target Objectness (`tgt_objectness`):* A tensor that will be set to 1.0 for the responsible boxes.
   - *Target Bounding Boxes (`tgt_xywh`):* A tensor that will store the GT bounding box coordinates.
   - *Target Classes (`tgt_cls`):* A tensor that holds the one-hot encoded class labels for each GT object.

2. **Grid Cell Identification:**

   For each ground truth object in each image, the center of the GT box (provided in normalized coordinates $[0, 1]$) is used to determine its corresponding grid cell. The cell indices are computed as:

   $$\text{grid\_x} = \lfloor x_{\text{gt}} \times W \rfloor, \quad \text{grid\_y} = \lfloor y_{\text{gt}} \times H \rfloor$$

   where x_{gt} and y_{gt} are the normalized center coordinates and W and H are the width and height of the grid.

3. **Box Evaluation and Selection:**

   Within the identified grid cell, all A predicted boxes are evaluated by computing their IoU with the GT box:

$$\text{IoU}(B_{\text{gt}}, B_{\text{pred}}) = \frac{\text{Area}(B_{\text{gt}} \cap B_{\text{pred}})}{\text{Area}(B_{\text{gt}} \cup B_{\text{pred}})}$$

   The predicted box with the highest IoU is selected as the candidate responsible for the GT object. If that box is already assigned to another object (as indicated by the `obj_mask`), the function searches for the next best candidate.

4. **Target Updates:**

   Once the best candidate is identified, the target tensors are updated:

   - *Objectness:* The corresponding element in `tgt_objectness` is set to 1.0.
   - *Bounding Box Coordinates:* The GT box [x_c, y_c, w, h] is assigned to `tgt_xywh` for that grid cell and box index.
   - *Class Label:* The appropriate entry in `tgt_cls` is set to 1.0 for the GT class.

   This update process ensures that the model is guided by the most accurate spatial and semantic supervision during training.

5. **Conflict Resolution:**

   If multiple GT boxes fall into the same grid cell or if a candidate box is already assigned, the function resolves these conflicts by re-evaluating the remaining candidates based on their IoU scores. If no unassigned box meets the criteria, the assignment for that GT object is skipped.

6.5 Image Reconstruction Analysis

In this section, we analyze the reconstructed images produced by our model to evaluate which features the encoder is able to preserve and to identify potential shortcomings for the object detection task. Our examination reveals that small

objects—specifically airplanes—tend to "disappear" in the reconstruction process. This is illustrated in Figure 6.5, where the global scene is well maintained, yet fine-grained details corresponding to small targets are lost.

The phenomenon appears to be caused by the model's emphasis on achieving global consistency. In doing so, the encoder's embedding space seems to "explain away" these small features as noise or insignificant details. Consequently, the resulting representations capture dominant, large-scale structures but fail to retain the subtle cues that are essential for accurately localizing small objects.

This analysis leads to the critical deduction that the features extracted by the encoder, in its current form, are not optimally suited for the object detection task. To address this issue, it becomes necessary to perform fine-tuning—or even full retraining—of the encoder with a specific focus on preserving fine details. Such an approach would require a significantly larger dataset to ensure that the encoder learns to capture both global and local features effectively, compared to scenarios where only the detection head is trained.

Overall, our findings suggest that enhancing the encoder through targeted fine-tuning and increased data diversity could substantially improve the detection performance for small objects. Future work should explore these avenues further, incorporating advanced data augmentation techniques and a larger volume of annotated data to better guide the encoder's learning process.

6.6 Experiments with Clay and Their Results

In our experiments with the Clay model, we compared two strategies regarding the encoder: one in which the Clay encoder was trained (unfrozen) and another in which the encoder was frozen during training. Analysis of the mAP curves revealed that the network was unable to correctly identify airplanes in either configuration. In the experiment where the encoder was trained, the model showed clear signs of overfitting—likely due to the limited dataset size or the high number of trainable parameters—which negatively impacted its generalization capability. Conversely, when the encoder was frozen, the number of trainable parameters was reduced, thereby mitigating overfitting; however, this configuration still did not succeed in

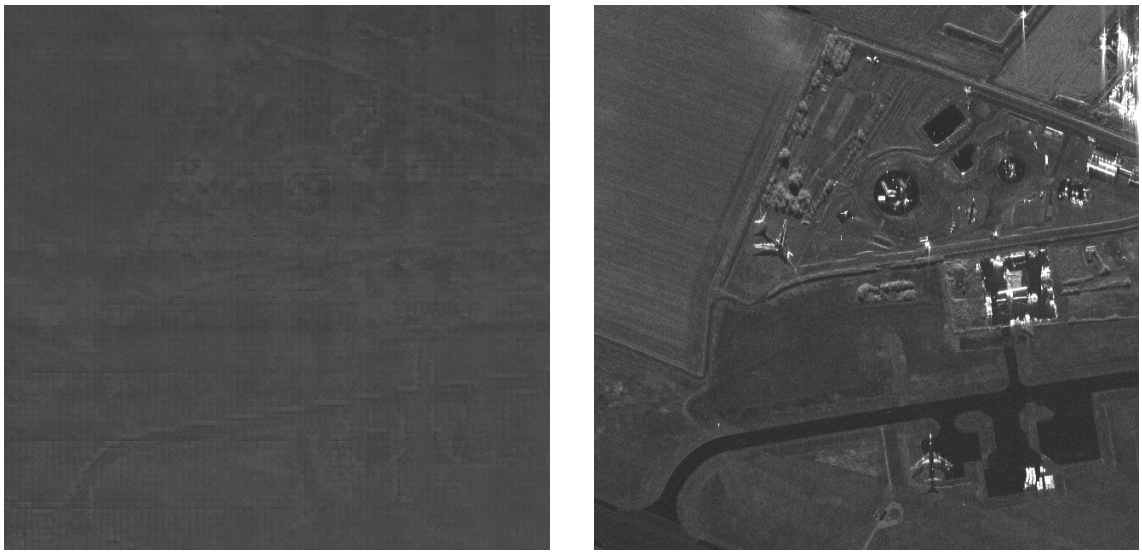


Figure 6.3: An example of the reconstruction, where the general features are preserved but small objects disappear.

detecting airplanes. Both experiments were conducted for a total of 500 epochs using the total loss function described in Section 6.3 and the hyperparameter configuration detailed in Section 6.2.1. For further insights, please refer to the accompanying graphs.

6.7 Alternative Backbone Evaluation: ResNet50

While the primary focus of this work is on extending Clay's specialized ViT-based encoder for object detection tasks, an alternative approach was explored by replacing the Clay encoder with a classical ResNet50 backbone. This section details the evaluation of ResNet50 as a feature extractor for our custom object detection head, discusses the experimental configurations, and compares its performance with the baseline.

Motivation

ResNet50 is widely adopted in computer vision due to its simplicity and extensive support across various frameworks. Although using ResNet50 means relinquishing the direct spatiotemporal embeddings provided by Clay's Vision Transformer, it offers a computationally less intensive alternative with lower memory demands. This alternative evaluation aimed to determine whether ResNet50 could achieve compet-

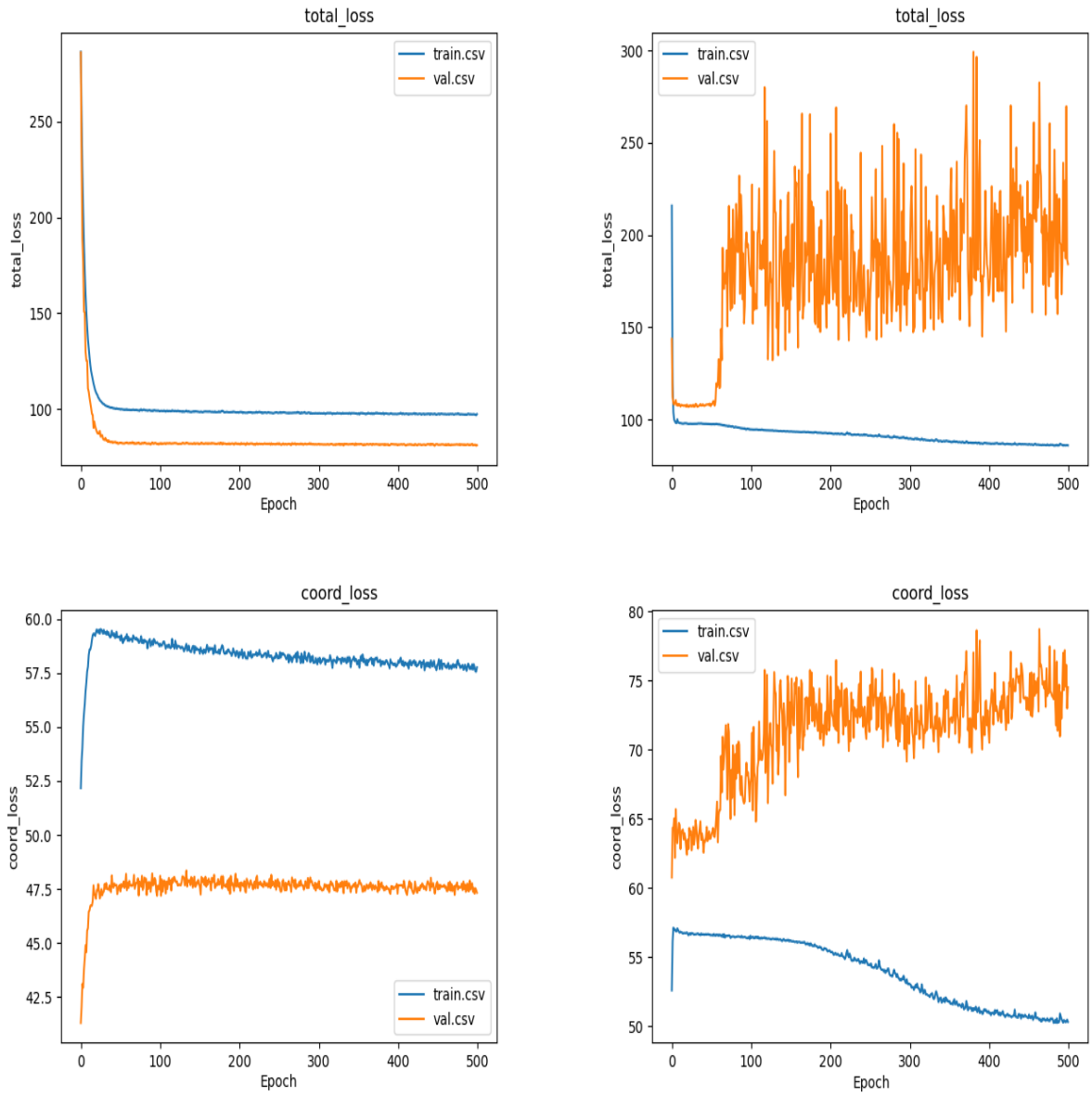


Figure 6.4: Compared results of Clay with the encoder trained and Clay with the encoder freezed. Coord Loss is the Coordinate Loss described in the Chapter 6.3.

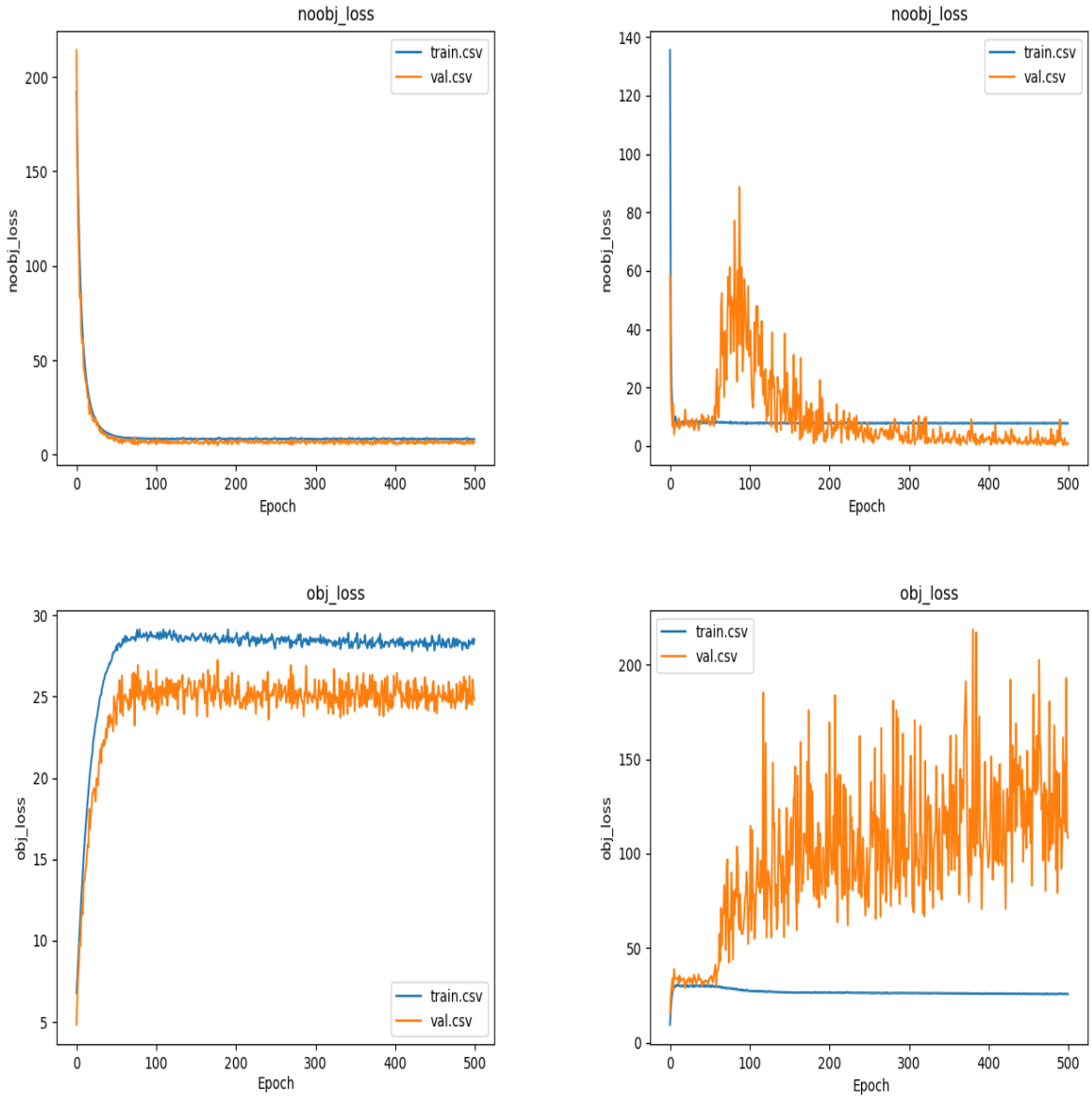


Figure 6.5: Compared results of Clay with the encoder trained and Clay with the encoder freezed. Noobj Loss and Obj Loss are the partial losses that compose the Objectness Loss described in the Chapter 6.3.

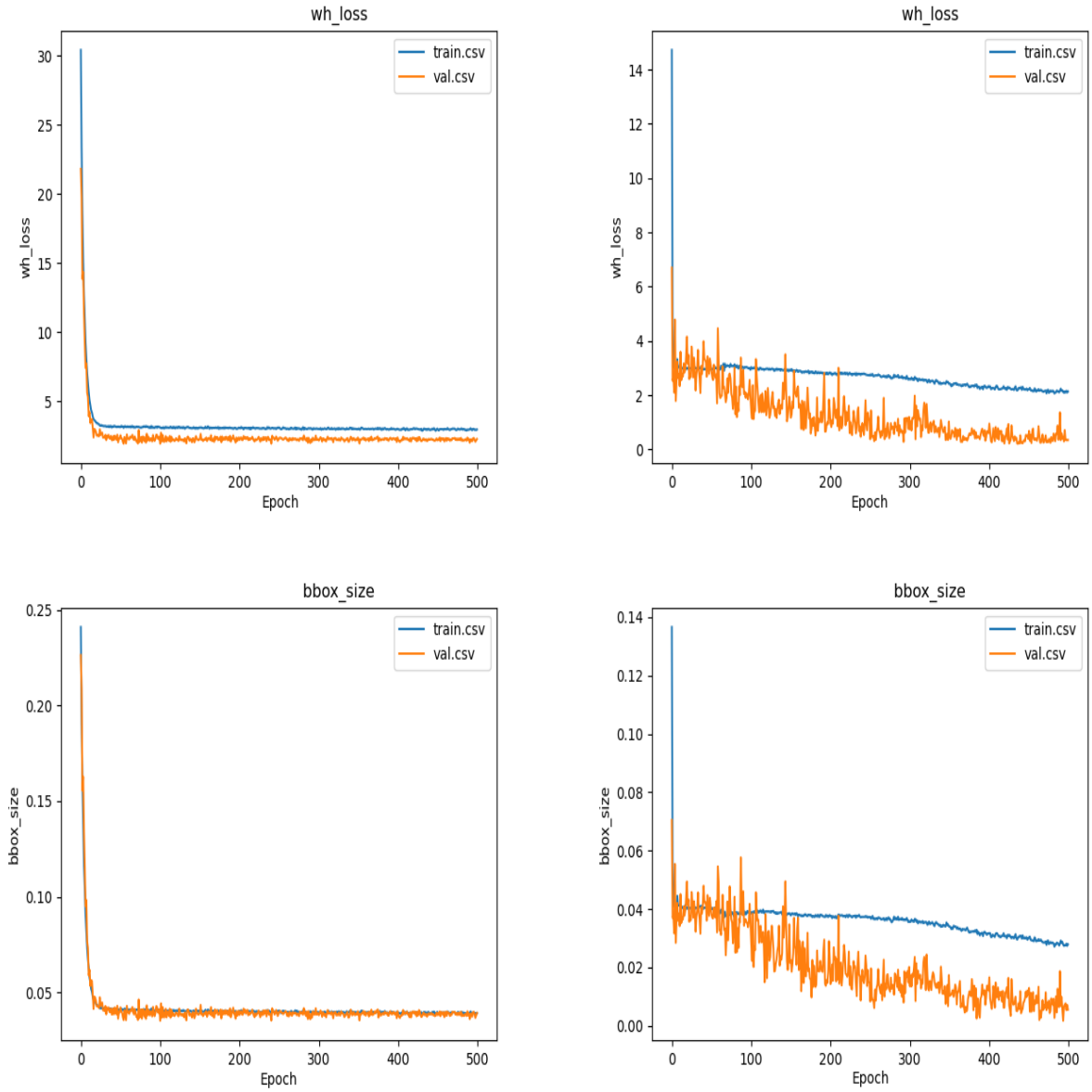


Figure 6.6: Compared results of Clay with the encoder trained and Clay with the encoder freezed. WH Loss represent the MSE loss used to penalize extremely large boxes. BBox Size show the evolution of the bounding box size over the epochs.

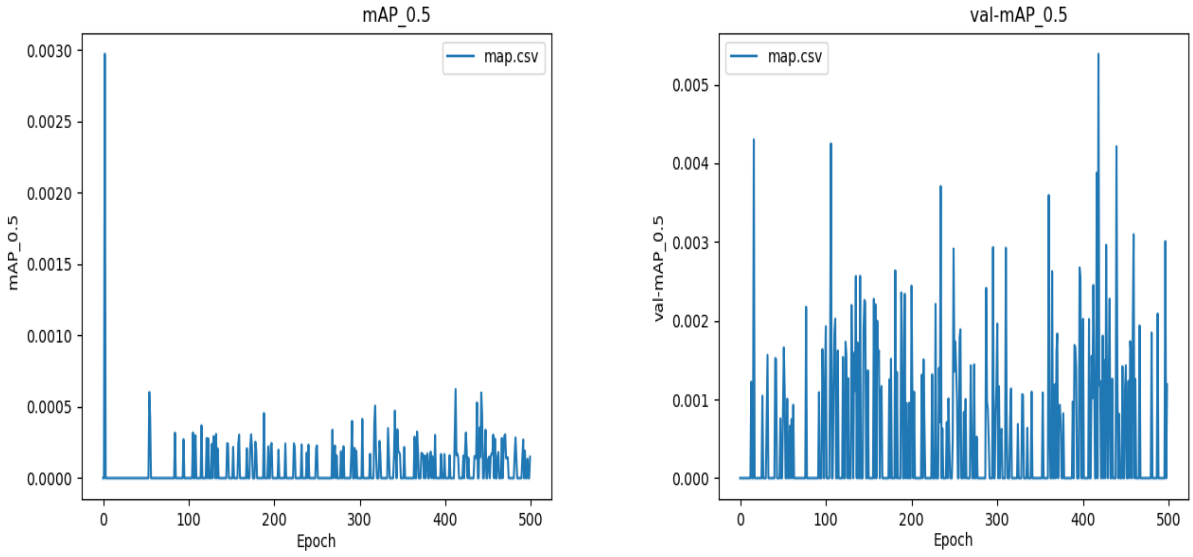


Figure 6.7: Compared results of Clay with the encoder trained and Clay with the encoder freezed.

itive detection performance, particularly in the context of bounding box regression and classification for detecting small objects like airplanes.

Experimental Configurations

Two primary experimental setups were evaluated with ResNet50:

1. **CIOU + MSE Loss Configuration**

   In this setup, the object detection head was trained for 500 epochs using a composite loss function that included:

   - *Complete Intersection over Union (CIoU) Loss:* This component captures the overlap between predicted and ground truth boxes, the centroid distance, and the aspect ratio consistency.
   - *Mean Squared Error (MSE) Loss for Width and Height:* This penalty term is applied to reduce deviations in the predicted box dimensions.

   The inclusion of MSE loss served to discourage the model from predicting overly large bounding boxes in an attempt to maximize the CIoU score, thereby promoting a more precise regression of box dimensions.

2. **CIOU-Only Configuration**

   In this experiment, the same training conditions were applied except that the

MSE penalty for the width and height was removed. The absence of the MSE component resulted in an initial tendency of the network to predict larger bounding boxes, as it primarily optimized for the CIoU metric.

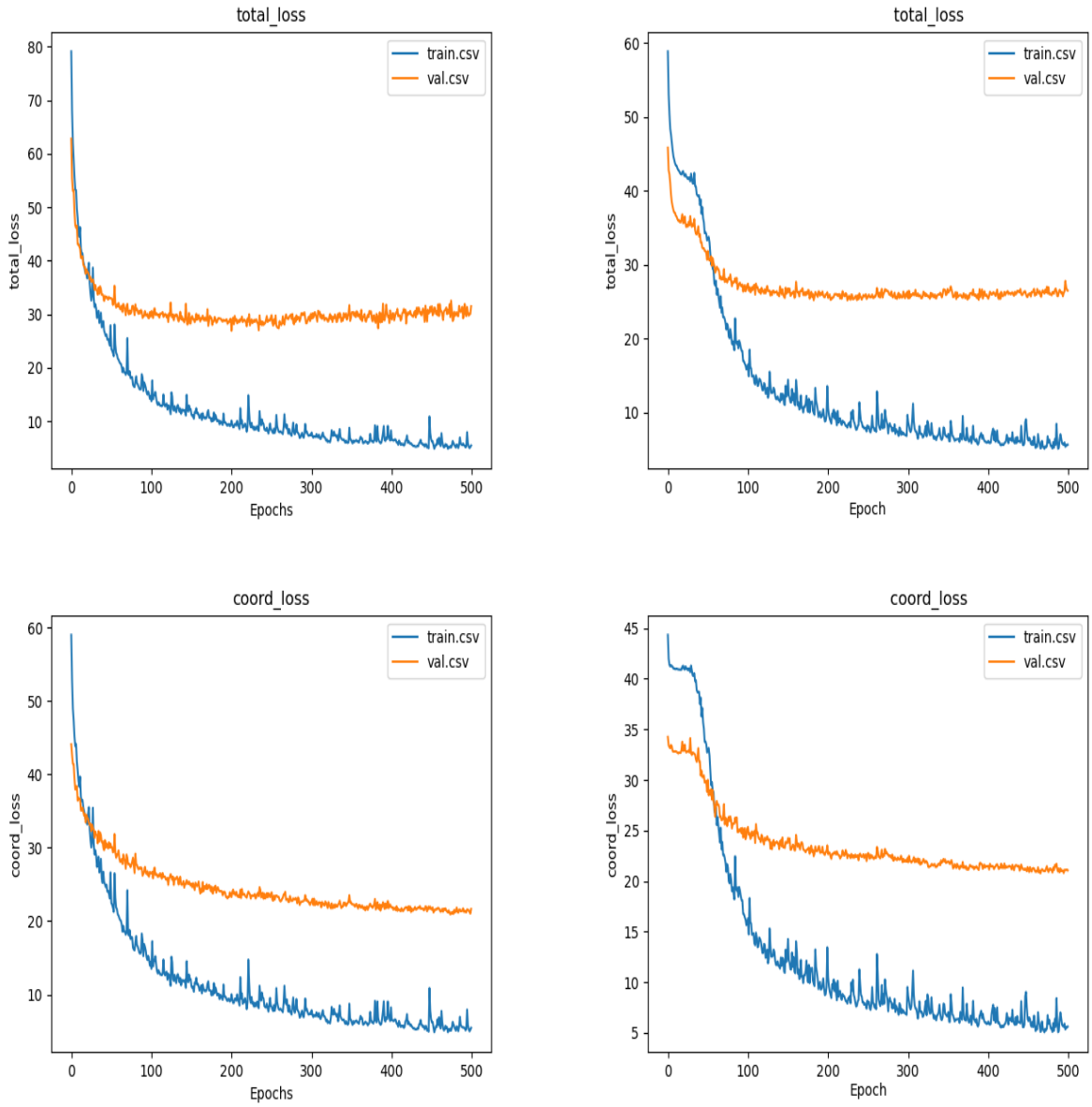


Figure 6.8: Compared results of using ResNet as feature extractor. Coord Loss is the Coordinate Loss described in the Chapter 6.3.

Additionally, two supplementary experiments were conducted:

- **Backbone Frozen:** Here, the ResNet50 backbone was kept frozen (pre-trained weights were not updated) during the 500-epoch training period. This

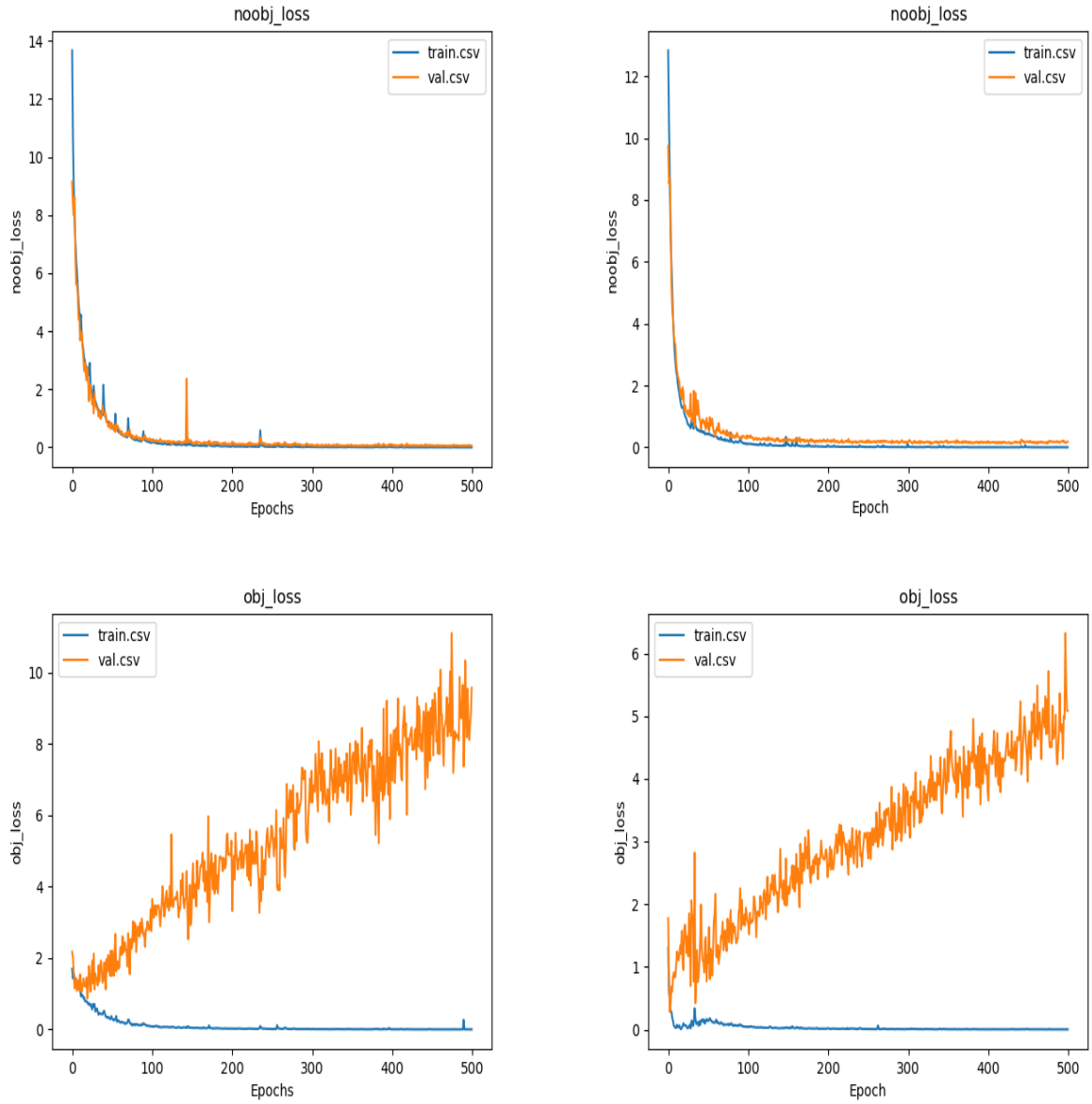


Figure 6.9: Compared results of using ResNet as feature extractor. Noobj Loss and Obj Loss are the partial losses that compose the Objectness Loss described in the Chapter 6.3.

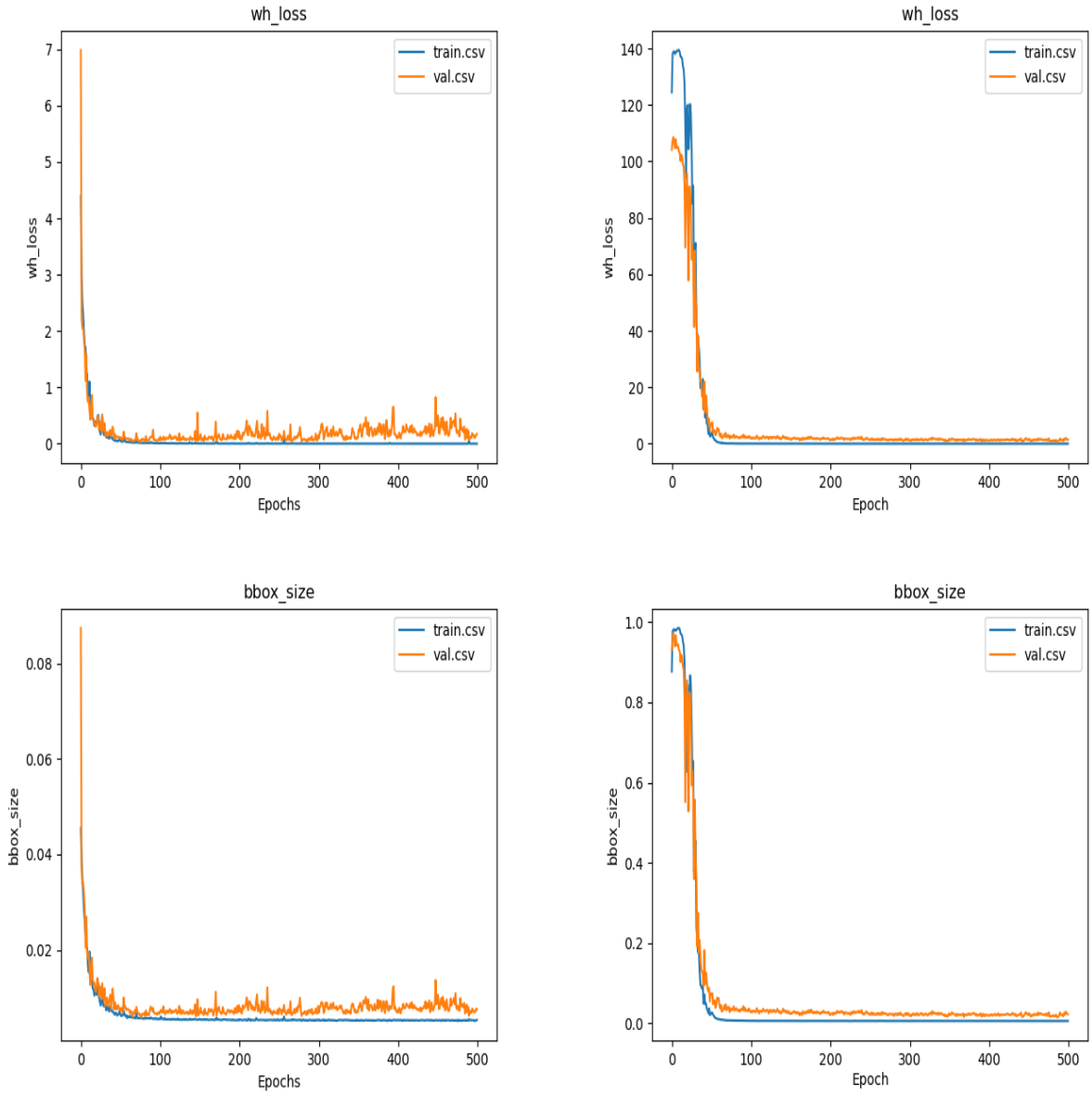


Figure 6.10: Compared results of using ResNet as feature extractor. WH Loss represent the MSE loss used to penalize extremely large boxes. BBox Size show the evolution of the bounding box size over the epochs.

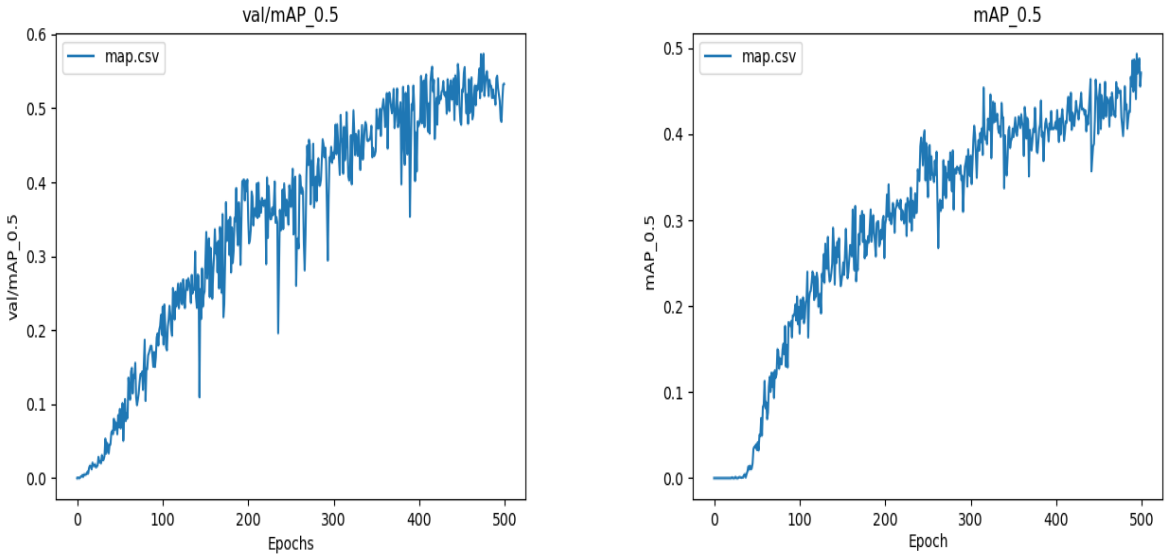


Figure 6.11: Compared results of using ResNet as feature extractor.

approach, while reducing the number of trainable parameters, resulted in sub-optimal performance since the backbone, originally pre-trained for classification, did not adapt well to the detection task.

- **Long Train with StepLR Scheduler:** In this configuration, the entire network was trained for 3000 epochs with a StepLR scheduler that reduced the learning rate by a factor of 0.1 every 1000 epochs. This extended training regimen led to better convergence and improved detection performance overall.

Results and Discussion

The experiments revealed several key findings:

- **Effect of MSE Loss:** The inclusion of the MSE loss in the CIOU + MSE configuration significantly helped in refining the bounding box dimensions, leading to faster convergence and higher mAP scores compared to the CIOU-only setup. From Figure 6.10, we can observe that in the CIoU-only configuration the network tends to predict increasingly larger bounding boxes. This behavior occurs because, in order to maximize the IoU, the network can simply enlarge the overlapping area between the predicted box and the ground truth. During the initial epochs, the predicted box centers are often inaccurate; if

a small bounding box were predicted with an incorrect center, the resulting intersection area would be zero, yielding an IoU of zero. Only after many epochs, once the network has learned to accurately position the center, does it begin to reduce the size of the bounding boxes.

- **Dynamic vs. Frozen Backbone:** Training with a dynamic (trainable) ResNet50 backbone outperformed the frozen-backbone configuration, indicating that adaptation of the feature extractor to the specific object detection task is crucial.

- **Extended Training Benefits:** The long training experiment with a StepLR scheduler demonstrated improved performance, suggesting that the network benefits from longer fine-tuning periods when using ResNet50.

Overall, although ResNet50 lacks the specialized spatiotemporal feature extraction capabilities of the Clay encoder, it still serves as a viable alternative for object detection when computational simplicity and broader framework support are priorities. However, achieving competitive performance with ResNet50 requires careful fine-tuning, appropriate loss function design, and extended training with adaptive learning rate scheduling.

6.8 Limitations and Future Works

Several challenges and potential improvements in applying Clay to SAR data have been identified:

- **Memory:** Large input images or high batch sizes can quickly exhaust GPU memory, particularly when incorporating transformer-based components.

- **Compute Requirements:** Transformers generally require more computational resources than CNNs, leading to increased compute costs.

- **Data Augmentation:** Implementing additional transformations—such as random crops, flips, color jitter, and domain-specific augmentations for Earth Observation—could enhance model robustness.

- **Dataset Variety:** A small or less diverse dataset may limit the model's generalization capabilities, especially when rare objects are underrepresented.
- **Number of Examples:** Increasing the number of labeled examples is essential to improve detection performance, particularly for small and infrequent objects.
- **New Clay Versions:** Future releases of foundation models like Clay (e.g., version 1.5) or alternative large-scale models might further improve feature representation quality and overall detection performance.

Chapter 7

Conclusions and Future Work

This thesis set out to explore object detection in ultra-high-resolution SAR data, addressing several research questions regarding the applicability of state-of-the-art detection models, the role of color information in CSI images, the impact of resolution and patch size, and the use of foundation models such as Clay.

7.1 Addressing the Research Questions

1. **Can standard OD models be effectively applied to SAR data?** Yes, our findings indicate that YOLOv8, with appropriate fine-tuning, is effective on SAR imagery despite being pretrained on a different domain (COCO). This underscores the transferability of optical object detection models to the SAR context.

2. **Do colorized CSI images improve detection performance?** No, contrary to initial expectations, Colorized Sub-aperture Images (CSI)—which encode angular backscattering information—did not yield measurable improvements in object detection performance compared to standard grayscale amplitude-only SAR (GRD) images. This could be attributed to the fundamentally different significance of color information in CSI images compared to natural optical imagery used in pre-training.

3. **Does ultra-high resolution improve object detection performance?** Yes, ultra-high resolution enhances the detection of small objects like air-

craft. Nevertheless, the trade-off is a substantial increase in computational cost, which could limit the practical deployment of certain models, especially transformer-based ones.

4. **Are hyperparameter choices, such as resolution and patch size, critical for detection?** Absolutely. Selecting optimal hyperparameters, particularly resolution and patch size, was found essential for accurately localizing small-scale objects. The choice of patch size directly influenced the contextual information available to the model, significantly impacting localization accuracy. Optimal configurations were empirically determined to balance accuracy and computational efficiency effectively.

5. **Is the use of foundation models like Clay advantageous for OD in SAR?** In our experiments, the foundation model Clay did not yield the expected benefits for object detection. Its pretraining was not well-suited to the SAR domain, and the necessity to retrain its encoder was impractical due to dataset limitations.

7.2 Future Work

While this thesis has provided valuable insights into the application of YOLOv8 on SAR data and highlighted key considerations for hyperparameter tuning and data representation, several avenues remain open for further exploration:

- **Enhancing Transformer Architectures:** Future research could investigate modifications to transformer-based models to better handle the unique characteristics of SAR data, possibly by integrating domain-specific pretraining strategies.

- **Exploring Alternative Foundation Models:** Given the challenges encountered with Clay, exploring other foundation models or developing hybrid architectures that combine convolutional and transformer-based approaches might yield improved performance.

- **Expanding the Dataset:** Increasing the volume and diversity of labeled

SAR data could allow for more extensive fine-tuning and may overcome some of the limitations observed in the current study.

- **Cost-Effective Deployment:** Investigating model compression and optimization techniques could help mitigate the computational costs associated with ultra-high-resolution processing.

Considering these insights, future research directions include exploring transformer-based detection heads similar to the DETR architecture, which would naturally align with the structure of foundation models like Clay. However, this approach necessitates considerably larger and more diverse datasets to avoid overfitting and fully realize the model's potential.

Further research could explore advanced data augmentation strategies tailored explicitly for SAR imagery, potentially alleviating some limitations imposed by dataset scarcity. Additionally, investigations into hybrid architectures, integrating the computational efficiency of CNN-based methods with the flexibility and global context modeling capabilities of Transformers, could offer promising directions for improving SAR object detection systems.

In summary, while standard CNN-based models such as YOLO demonstrate reliable performance on SAR imagery, harnessing the full potential of transformer-based architectures and foundation models will require larger datasets, extensive computational resources, and careful tuning of hyperparameters. The findings of this thesis thus offer practical guidelines and underscore future directions for developing robust, accurate, and computationally feasible SAR object detection systems, which are increasingly vital for environmental monitoring, infrastructure management, and global security applications.

List of Figures

List of Tables

Bibliography

[1] Thomas Ager. The Essentials of SAR: A Conceptual View of Synthetic Aperture Radar. Independently published, August 14, 2021. ISBN: 979-8512864487.

[2] Bochkovskiy Alexey, Wang Chien-Yao, and Liao Hong-Yuan Mark. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: arXiv preprint arXiv:2004.10934 (2020). DOI: doi.org/10.48550/arXiv.2004.10934.

[3] Wang Chien-Yao, Bochkovskiy Alexey, and Liao Hong-Yuan Mark. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: arXiv preprint arXiv:2207.02696 (2022). DOI: doi.org/10.48550/arXiv.2207.02696.

[4] Li Chuyi et al. "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications". In: arXiv preprint arXiv:2209.02976 (2022). DOI: doi.org/10.48550/arXiv.2209.02976.

[5] Reis Dillon et al. "Real-Time Flying Object Detection with YOLOv8". In: arXiv preprint arXiv:2305.09972 (2023). DOI: doi.org/10.48550/arXiv.2305.09972.

[6] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: arXiv preprint arXiv:2010.11929 (2021). DOI: doi.org/10.48550/arXiv.2010.11929.

[7] Clay Foundation. Clay Foundation Model. URL: https://clay-foundation.github.io/model/index.html.

[8] ICEYE. Actionable intelligence through Earth Observation. URL: https://www.iceye.com/.

[9] ICEYE. Amplitude Image - Ground Range Detected. URL: https://sar.iceye.com/5.0/productFormats/grd/#binary-representation.

[10] ICEYE. Colorized Sub-aperture Image. URL: https://sar.iceye.com/5.2.4/productFormats/csi/.

[11] ICEYE. Imaging Modes. URL: https://sar.iceye.com/6.0.1/productspecification/imagingmodes/.

[12] Redmon Joseph and Farhadi Ali. "YOLO9000: Better, Faster, Stronger". In: arXiv preprint arXiv:1612.08242 (2016). DOI: doi.org/10.48550/arXiv.1612.08242.

[13] Redmon Joseph and Farhadi Ali. "YOLOv3: An Incremental Improvement". In: arXiv preprint arXiv:1804.02767 (2018). DOI: doi.org/10.48550/arXiv.1804.02767.

[14] Redmon Joseph and Angelova Anelia. "Real-Time Grasp Detection Using Convolutional Neural Networks". In: arXiv preprint arXiv:1412.3128 (2015). DOI: doi.org/10.48550/arXiv.1412.3128.

[15] Redmon Joseph et al. "You Only Look Once: Unified, Real-Time Object Detection". In: arXiv preprint arXiv:1506.02640 (2016). DOI: doi.org/10.48550/arXiv.1506.02640.

[16] S. Khan et al. A Guide to Convolutional Neural Networks for Computer Vision. Synthesis Lectures on Computer Vision. Morgan & Claypool Publishers, 2018, pp. 1–8.

[17] Ultralytics. YOLO. URL: https://docs.ultralytics.com/.

[18] Wikipedia. Description of the QGIS software environment. URL: https://it.wikipedia.org/wiki/QGIS.

[19] Aoran Xiao et al. "Foundation Models for Remote Sensing and Earth Observation: A Survey". In: arXiv preprint arXiv:2410.16602 (2024). DOI: doi.org/10.48550/arXiv.2410.16602.

[20] YOLOv8. YOLO. URL: https://yolov8.org/yolov8-architecture/.

Appendix A

Additional Information and Graphics for YOLO experiments

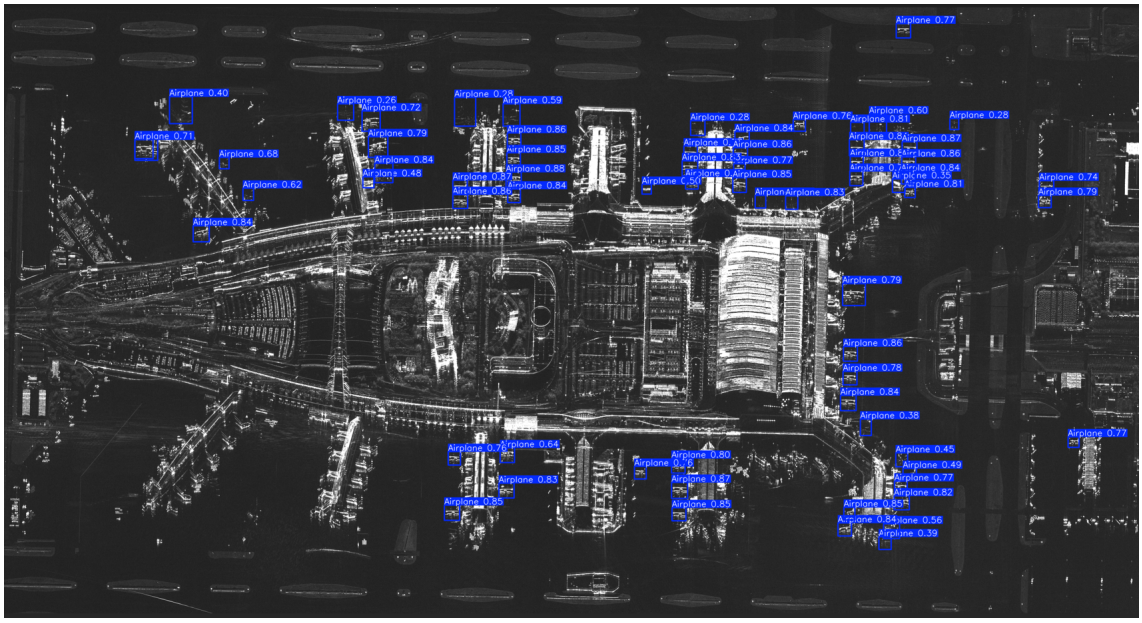


Figure A.1: Example of the inference result using YOLO.

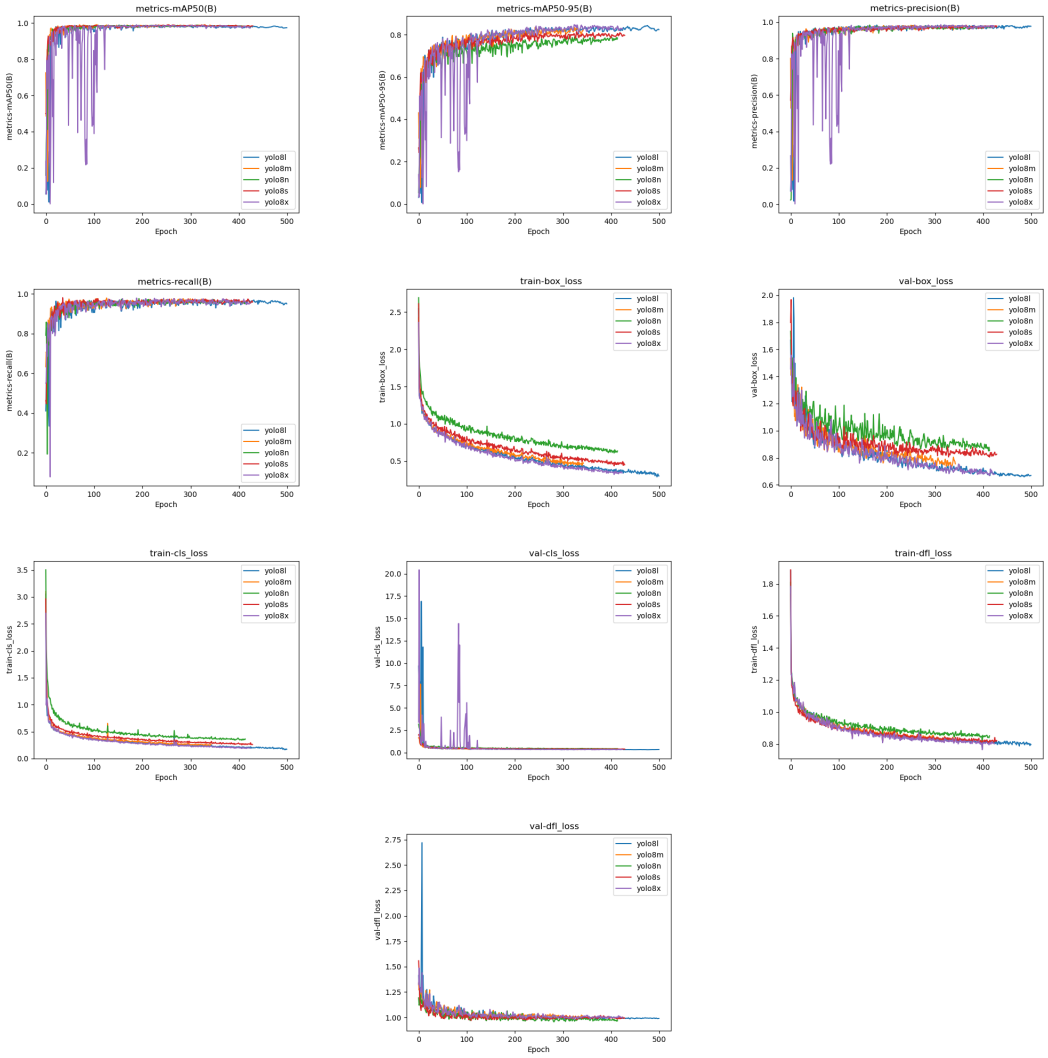


Figure A.2: Model complexity analysis results for grayscale images.

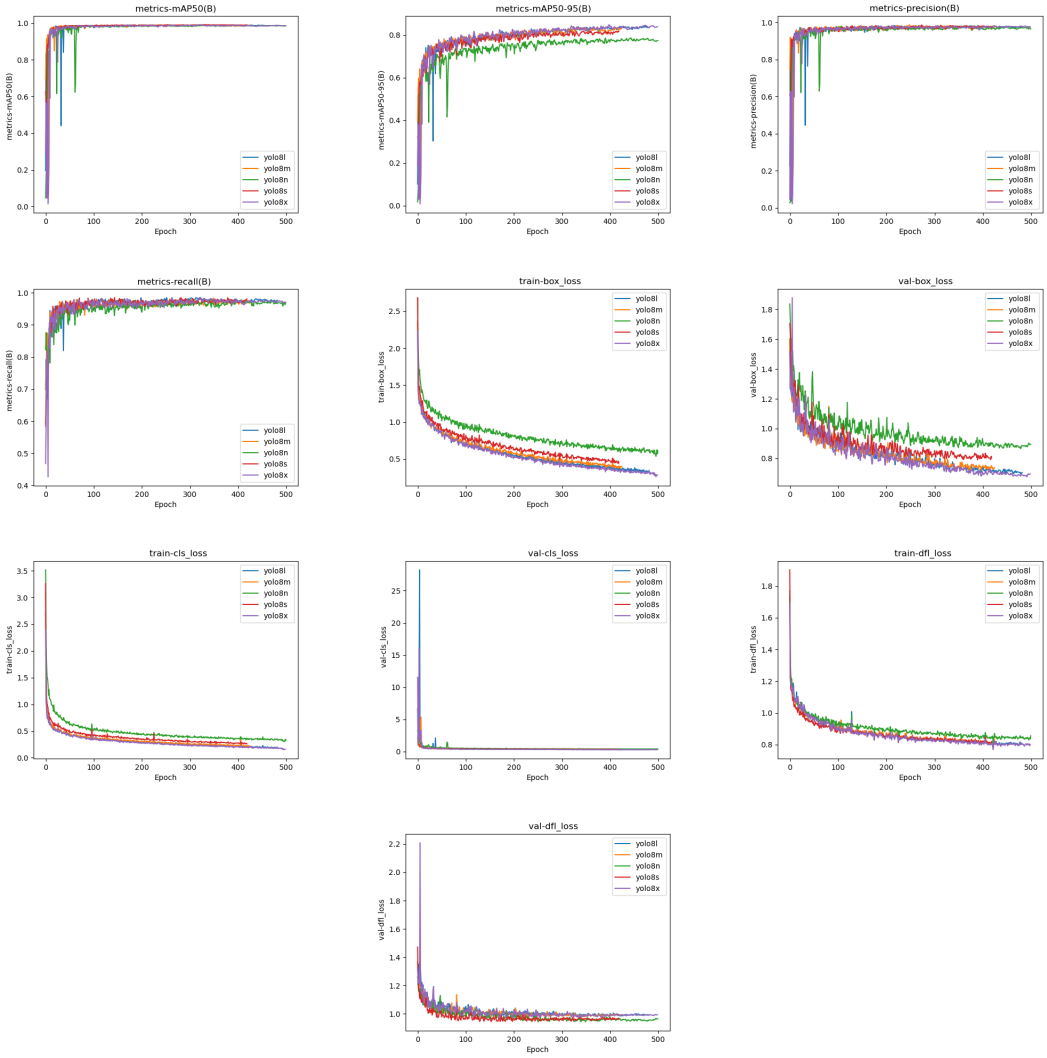


Figure A.3: Model complexity analysis results for color images.

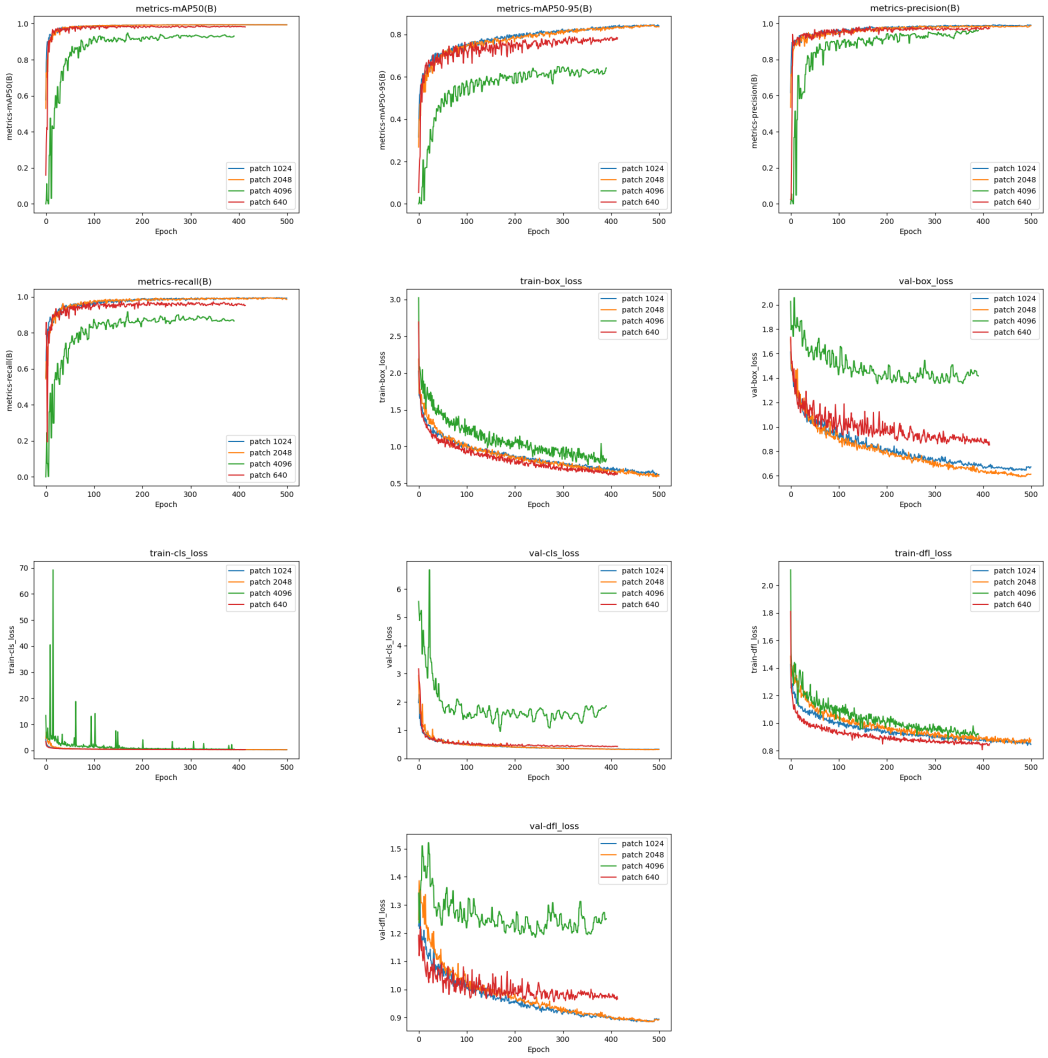


Figure A.4: Impact of Spatial Context results.

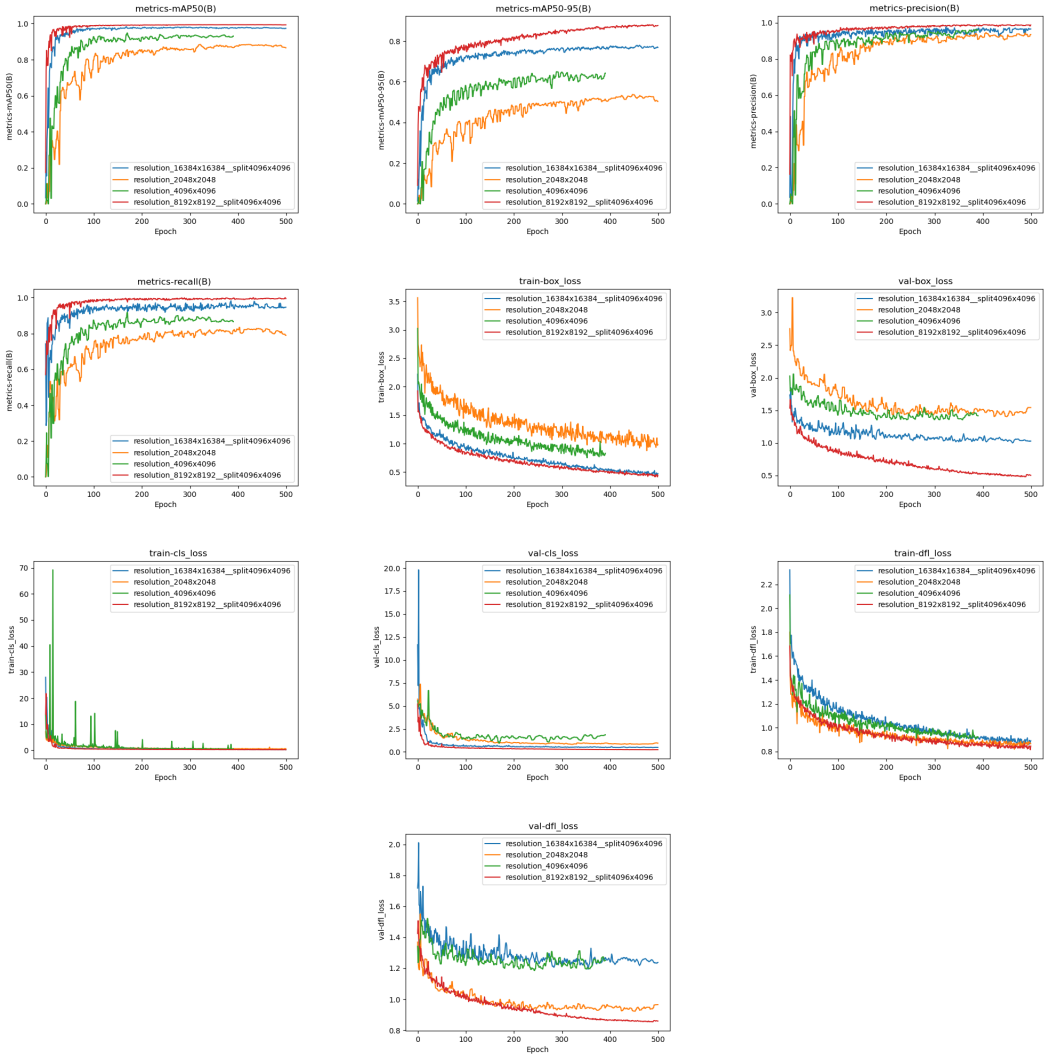


Figure A.5: Impact of Different Resolutions results.

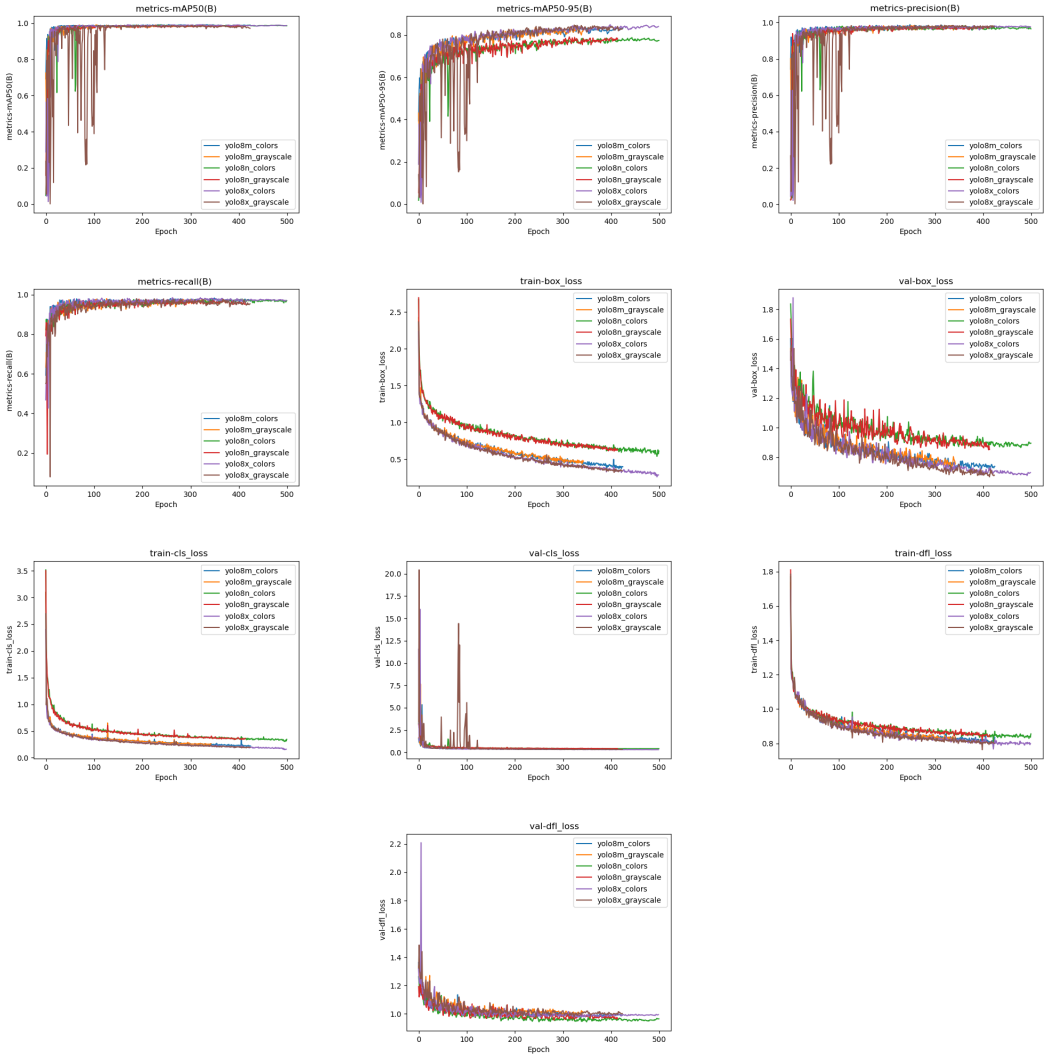


Figure A.6: Color versus Grayscale results.

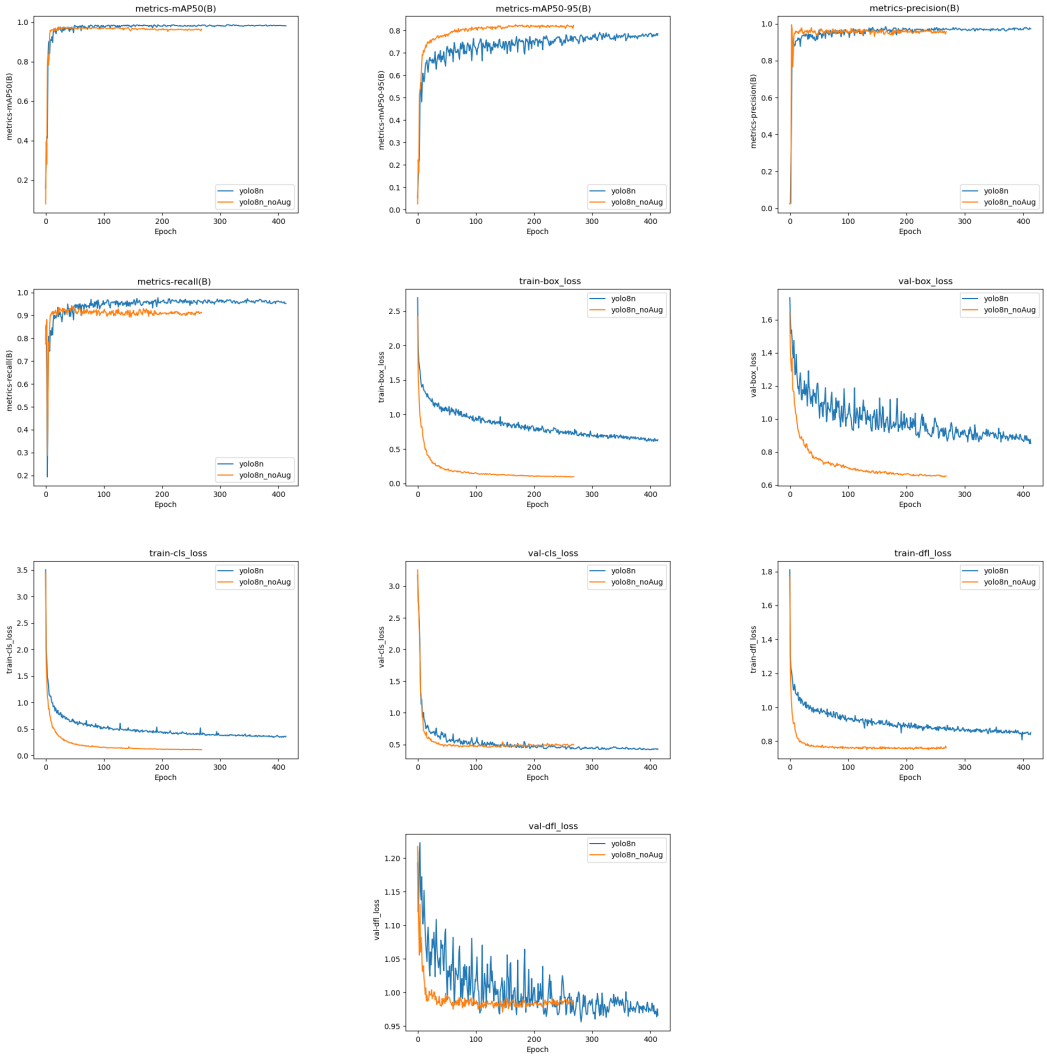


Figure A.7: Color versus Grayscale results.

Appendix B

Results and Graphics for Clay

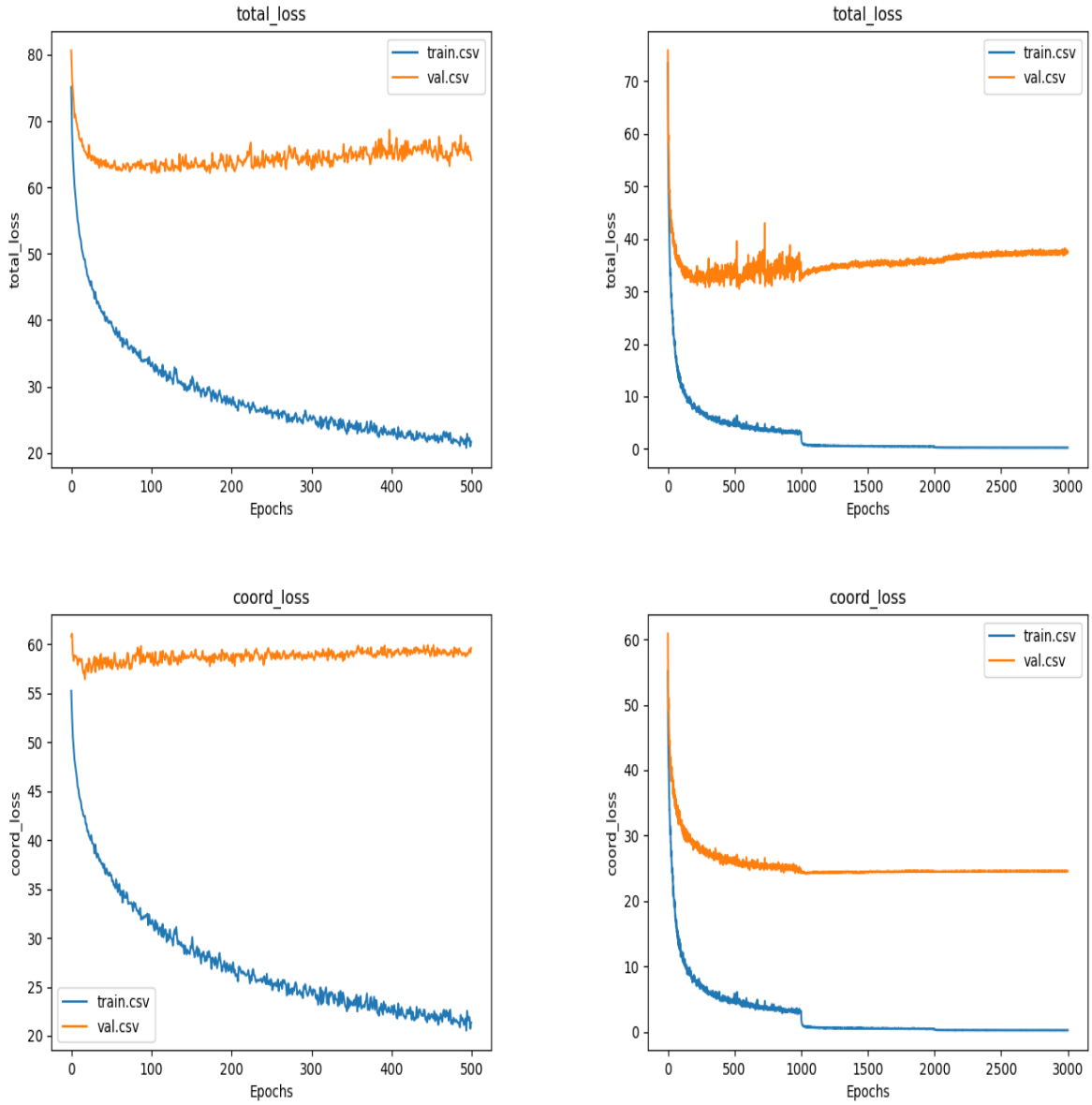


Figure B.1: Freezed ResNet on the left figure. Long training using Step Learning rate on the right figure. Both were trained using the training setup described in the Chapter 6.

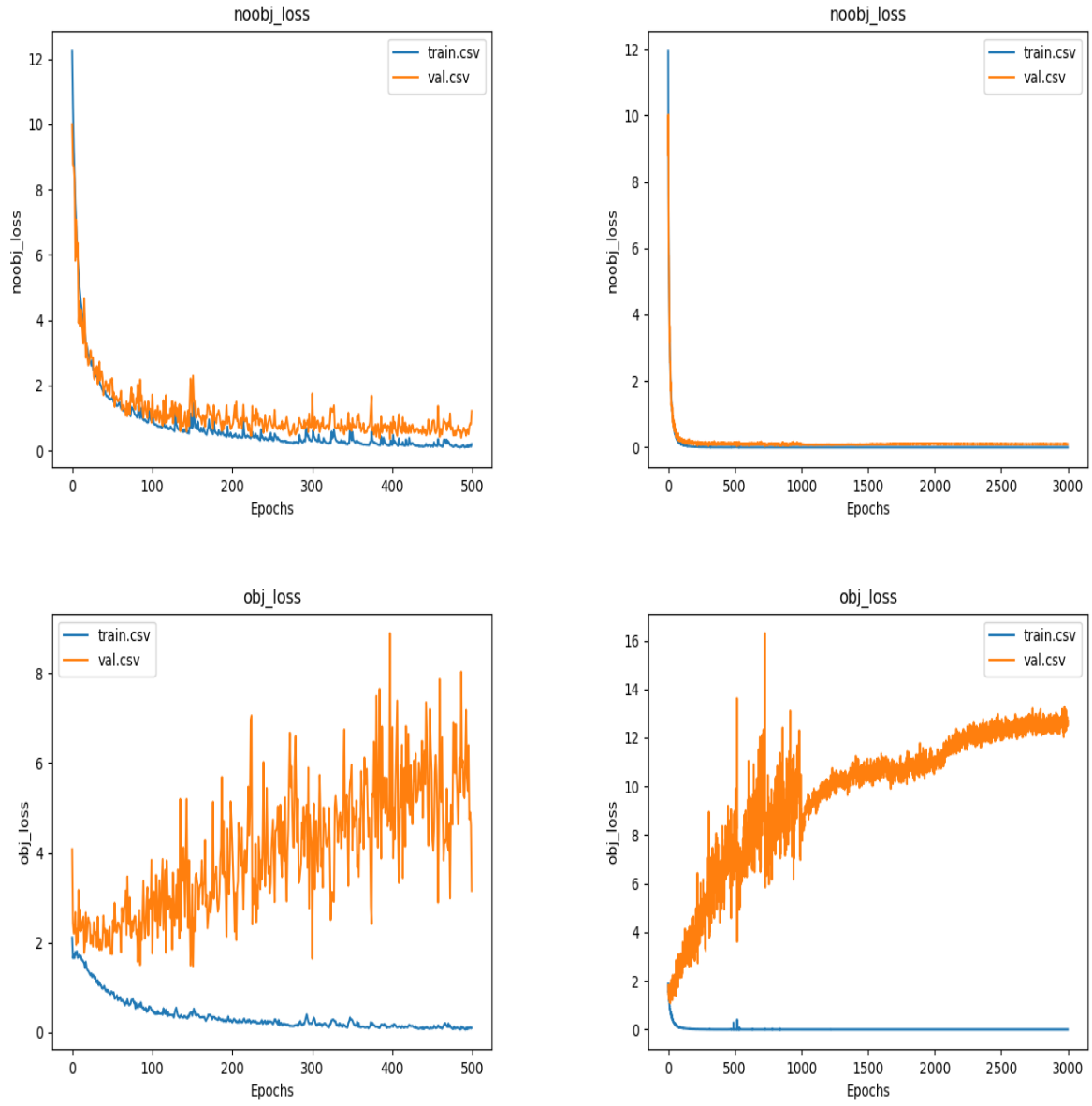


Figure B.2: Freezed ResNet on the left figure. Long training using Step Learning rate on the right figure. Both were trained using the training setup described in the Chapter 6.

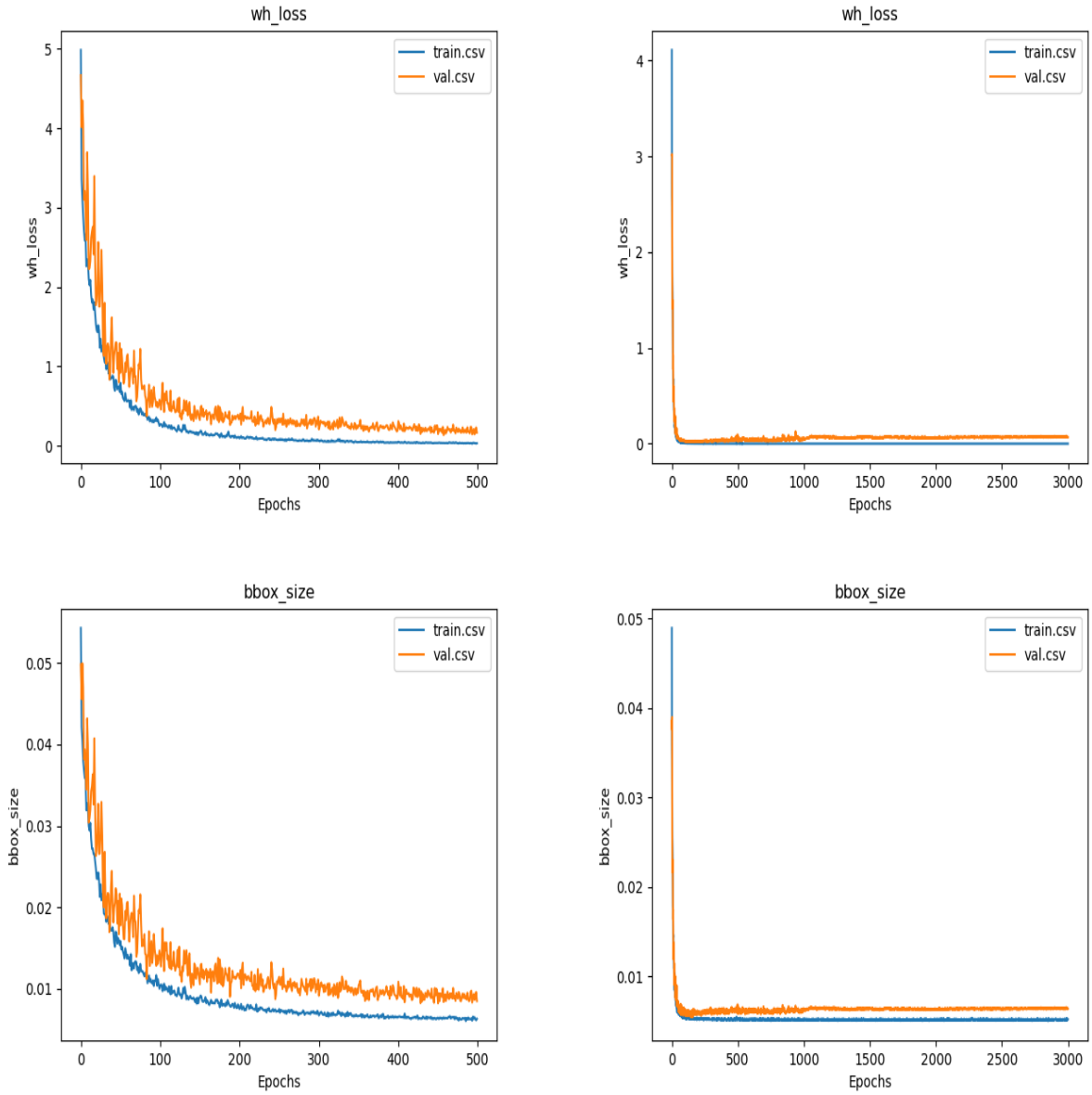


Figure B.3: Freezed ResNet on the left figure. Long training using Step Learning rate on the right figure. Both were trained using the training setup described in the Chapter 6.

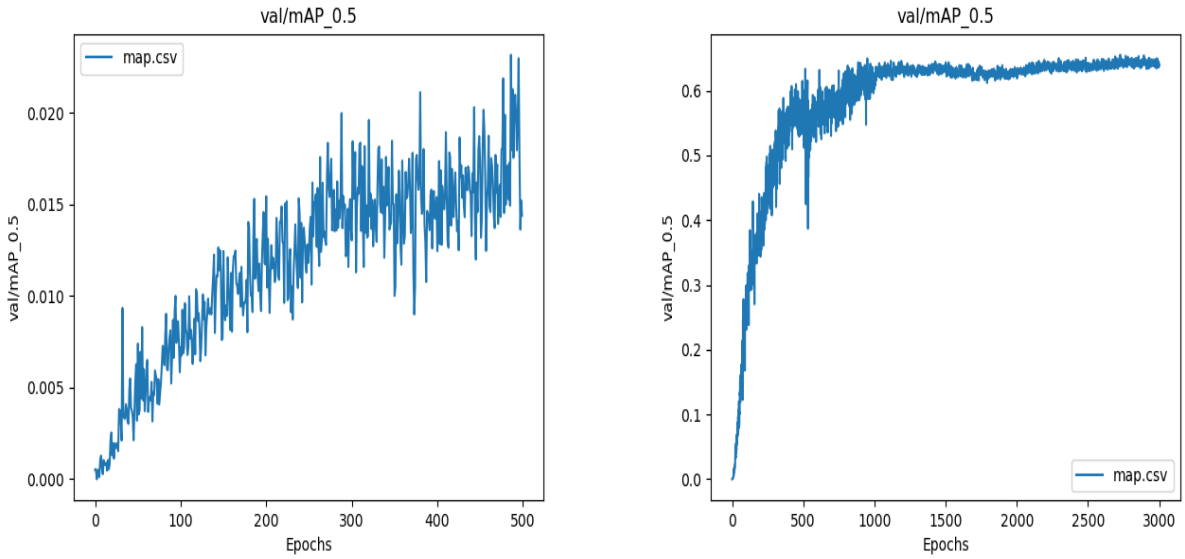


Figure B.4: Freezed ResNet on the left figure. Long training using Step Learning rate on the right figure. Both were trained using the training setup described in the Chapter 6.

Appendix C

Other Supplementary Material

```python
def clip_to_distribution(image):
    """
    Clips the values of an image based on the mean and standard
       deviation.
    """
    img_mean = np.nanmean(image)
    img_std = np.nanstd(image)
    min_val = math.floor(img_mean - 4 * img_std)
    max_val = math.floor(img_mean + 4 * img_std)
    return min_val, max_val


def convert_float_to_8(image, min_val=None, max_val=None):
    """
    Converts a float image to an 8-bit image for visualization.
    """
    if min_val is None or max_val is None:
        min_val = np.nanmin(image)
        max_val = np.nanmax(image)
    scale = 255 / (max_val - min_val)
    image = np.clip(image, min_val, max_val)
    return ((image - min_val) * scale).astype(np.uint8)
```

Listing C.1: Python code for processing GRD images

```python
def process_grd_image(input_path, output_path, calibration_factor):
    """
    Processes a GRD image from a .tif file and saves it as a PNG
       with values optimized for the human eye.
    """
    try:
        # Load GRD image
        dataset = gdal.Open(input_path)
        band = dataset.GetRasterBand(1)
        grd = band.ReadAsArray()
        if grd is None:
            raise FileNotFoundError(f"File {input_path} not found
               or unreadable.")

        # Apply Log scale
        grd_log = 10*np.log10(grd**2*calibration_factor)
        # Clipping based on mean and standard deviation
        min_val, max_val = clip_to_distribution(grd_log)

        # Convert to 8-bit
        grd_8_bit = convert_float_to_8(grd_log, min_val, max_val)

        # Save the image as PNG
        img = Image.fromarray(grd_8_bit, mode='L')
        img.save(output_path, format='PNG')
        print(f"Image saved in PNG format: {output_path}")
    except Exception as e:
        print(f"Error during the image processing: {e}")
```

Listing C.2: Python code for processing GRD images

```python
def build_target(bbox_out, cls_out, gt_boxes_batch, gt_labels_batch
    , H, W, A, C):
    """
    Create the target tensors for the loss.

    Args:
        bbox_out: tensor of bounding box predictions [B, 5*A, H, W]
        cls_out: tensor of class predictions [B, A*C, H, W]
```

```
        gt_boxes_batch: list of len B, each element is a tensor [num_obj, 4] in (xc, yc, w, h)
        gt_labels_batch: list of len B, each element is a tensor [num_obj] with int class
        H: high of the grid
        W: width of the grid
        A: number of bounding boxes per cell
        C: number of classes

    Returns:
        obj_mask: [B, H, W, A] bool, True for responsible box
        noobj_mask: [B, H, W, A] bool, True for not responsible box
        tgt_objectness: [B, H, W, A] float, target objectness
        tgt_xywh: [B, H, W, A, 4] float, target bounding box
        tgt_cls: [B, H, W, A, C] float, target class
    """
    device = bbox_out.device
    B = bbox_out.size(0)

    # Reshape
    bbox_out = bbox_out.view(B, A, 5, H, W).permute(0, 3, 4, 1, 2).contiguous()  # [B, H, W, A, 5]
    cls_out = cls_out.view(B, A, C, H, W).permute(0, 3, 4, 1, 2).contiguous()    # [B, H, W, A, C]

    # Inizialize the target tensors
    obj_mask = torch.zeros((B, H, W, A), dtype=torch.bool, device=device)
    noobj_mask = torch.ones((B, H, W, A), dtype=torch.bool, device=device)
    tgt_objectness = torch.zeros((B, H, W, A), device=device)
    tgt_xywh = torch.zeros((B, H, W, A, 4), device=device)
    tgt_cls = torch.zeros((B, H, W, A, C), device=device)

    matched_bboxes = [[] for _ in range(B)]

    for b in range(B):
        gt_boxes = gt_boxes_batch[b]     # [num_obj, 4]
        gt_labels = gt_labels_batch[b]   # [num_obj]
```

```
        num_obj = gt_boxes.size(0)

        for obj in range(num_obj):
            gt_box = gt_boxes[obj]       # [4]
            gt_label = gt_labels[obj]     # scalar int

            # Determine the cell where the center of the GT falls
            grid_x = int(gt_box[0] * W)
            grid_y = int(gt_box[1] * H)

            # Clamp
            grid_x = max(0, min(W - 1, grid_x))
            grid_y = max(0, min(H - 1, grid_y))

            # Extract all predicted bounding boxes in the cell [grid_y, grid_x]
            pred_boxes = bbox_out[b, grid_y, grid_x, :, :4]  # [A, 4]

             # Find the index of the box with the maximum IoU
            best_idx = find_responsible_box(pred_boxes, gt_box)  # scalar int

            # Check if the box is already assigned to another GT
            # This avoids assigning multiple GTs to the same box
            if obj_mask[b, grid_y, grid_x, best_idx]:
                ious = box_iou(pred_boxes, gt_box)  # [A]
                # Set -inf for the boxes already assigned
                ious[obj_mask[b, grid_y, grid_x, :]] = -1
                best_iou, best_idx = torch.max(ious, dim=0)
                if best_iou == -1:
                    # All boxes are already assigned
                    continue

            # Assign the box as responsible
            obj_mask[b, grid_y, grid_x, best_idx] = True
            noobj_mask[b, grid_y, grid_x, best_idx] = False

            # Set the target
```

```
77              tgt_objectness[b, grid_y, grid_x, best_idx] = 1.0
78              tgt_xywh[b, grid_y, grid_x, best_idx, :] = gt_box
79              tgt_cls[b, grid_y, grid_x, best_idx, gt_label] = 1.0
80
81              # Calculate the IoU between prediction "best_idx" and
                    gt_box
82              ious = box_iou(pred_boxes, gt_box)  # shape [A]
83              match_iou = ious[best_idx].item()
84
85              # Save the matched boxes for logging
86              matched_bboxes[b].append({
87                  "cell_xy": (grid_y, grid_x),
88                  "anchor_idx": int(best_idx),
89                  "gt_box": gt_box.detach().cpu().tolist(),
90                  "pred_box": pred_boxes[best_idx].detach().cpu().
                        tolist(),
91                  "gt_label": gt_label,
92                  "iou": match_iou
93              })
94
95      return obj_mask, noobj_mask, tgt_objectness, tgt_xywh, tgt_cls,
            matched_bboxes
```

Listing C.3: Python code for Matching