Computer Vision for Space Sustainability and Resilient Satellite Internet

by

Wenxuan Zhang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master's of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The increasing adoption of satellite internet with low-Earth-orbit (LEO) satellites in mega-constellations provides ubiquitous connectivity to rural and remote areas. At the same time, the numerous space objects, dynamic space environment, complex atmospheric conditions, and system vulnerability to weather conditions have brought two critical challenges to maintaining space sustainability and resilient satellite internet: the first challenge is to detect space objects timely and effectively and the second challenge is to detect adverse weather conditions which disrupt the space-ground link conditions between LEO satellites and ground stations.

This paper addresses the first challenge by proposing four new deep learning (DL) models, GELAN-ViT, GELAN-ViT-Reduced, GELAN-RepViT-Extended, and GELAN-RepViT, incorporating a Vision Transformer (ViT) into the Generalized Efficient Layer Aggregation Network (GELAN) architecture. These models outperform the state-of-the-art YOLOv9-t in terms of mean average precision (mAP) and computational costs.

This paper addresses the second challenge by proposing an efficient transfer learning (TL) method that can enable a ground component to locally detect representative weather-related conditions. The method can detect snow, wet, and other conditions resulting from adverse and typical weather events and shows superior performance compared to the typical DL methods, such as YOLOv7, YOLOv9, Faster R-CNN, and R-YOLO. Our TL method also shows the advantage of being generalizable to various scenarios.

Acknowledgments

I express my deepest gratitude to my supervisor, Professor Peng Hu, for his expert guidance, unwavering support, and invaluable assistance throughout my research journey. His guidance at every step of this research was instrumental in shaping its direction and outcomes. I am particularly thankful for his supervision during my academic research, as well as his significant feedback and review of my academic writing.

I am also grateful to my family for their patience and support throughout my journey. Their belief in my passion has been a constant source of inspiration and motivation for my continued studies.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AP$_i$ Average Precision at a i% IoU threshold 11

CL Confidence Level 18

CNN Convolutional Neural Network 4

CV computer vision 1

DL deep learning 3

FPN Feature Pyramid Network 4

GELAN Generalized Efficient Layer Aggregation Network 3

HPC High-Performance Computing 8

IoU Intersection over Union 11

LEO low-Earth-orbit 1

LiDAR Light Detection and Ranging 2

mAP mean Average Precision 9

mAP50 mAP at a 50% IoU threshold 9

mAP50:95 mAP across a range of 50% to 95% IoU 9

ML machine learning 2

PGI Programmable Gradient Information 4

SOD satellite object detection 1

SODD SOD Dataset 9

SSD Single-Shot Detection 21

TL transfer learning 2

ViT Vision Transformer 3

VOC Visual Object Classes 4

YOLACT You Only Look At CoefficienTs 22

YOLO You Only Look Once 3

List of Symbols

α Scaling factor for the YOLO-based composite loss function 27

C Number of object classes 11

D_{ST} Simplified Target Domain 23

D_S Source Domain 23

D_T Target Domain 23

F_g Local Features 5

F_l Local Features 5

FP False Positive 27

$F(I;W)$ Neural network function with input I and weights W 23

$\nabla L(D_{ST};W)$ Gradient of the loss L with respect to the input domain D_{ST} and weights W 24

L_{box} Box regression loss 23

L_{cls} Classification loss 23

L_{mask} Mask loss 23

L Loss function 23

L^* Loss of the customized core classification for our specific task 30

L_{YOLACT} Loss for YOLACT 23

$MMD(D_S, D_T)$ Maximum Mean Discrepancy between source and target domains 23

$\mathbf{M_{ResNet50+FC}}$ Proposed ResNet50-based model 25

$N_{correct}$ Number of images that are correctly classified 27

nl Number of layers in the model 24

N Total number of images 27

O_I Object of Interest 23

OPT_T Optimal weight for the Target Domain 23

$p_i(r)$ Precision-recall curve for IoU threshold i 11

TP True Positive 27

Chapter 1

Introduction

Advancements in technology have enabled machines to interpret visual data with unprecedented accuracy, paving the way for computer vision (CV) applications in satellite operations. CV is a field of artificial intelligence focused on enabling computers to understand and process visual information from the world, encompassing key tasks such as image classification, object detection, and object segmentation. Image classification involves assigning labels or categories to images or objects within images. Object detection identifies objects and determines their locations within the image. Object segmentation provides pixel-level detail by partitioning an image into regions, enabling tasks of terrain mapping.

With the rapid increase of advanced low-Earth-orbit (LEO) satellites in large constellations, space assets are set to play a crucial role in the future, providing global internet access and space relay systems for deep space missions. However, managing thousands of LEO satellites while ensuring safe operations, space sustainability, and space situational awareness is becoming a significant challenge. For example, the increasing risk of collisions between LEO satellites and space objects can cause a significant amount of space debris of various sizes and threaten the safe operations of spacecraft and the space environment. Additionally, the performance of satellite internet systems is highly dependent on weather conditions, which can disrupt the operations of ground components critical to these networks, including antennas at emerging ground stations and user terminals.

A key challenge toward a sustainable satellite internet system is the lack of satellite object detection (SOD) solution for collision assessment and avoidance. In this case, satellites, particularly LEO satellites, should be able to efficiently detect other satellite objects within a certain field of view with high precision and low latency. In addition, considering the constrained hardware capacity and energy on an LEO satellite, the SOD task needs to

be as resource-efficient as possible. Currently, there is no good solution to this challenge. Despite the ease of manipulation and management, ground radar or laser telescope facilities for satellite tracking require some tolerance of latency introduced by the contact times and the transmission of the detection results, not to mention the high cost of the facilities. The Light Detection and Ranging (LiDAR) sensors have a limited detection range and are susceptible to atmospheric conditions, so ground detection using current LiDAR sensors is hardly feasible. Therefore, to reduce the latency of SOD tasks, we need onboard solutions on the satellites so they can detect the objects individually. Such onboard sensing is expected to produce results in near real-time. This leads to the question of what sensing mechanism is suitable for small satellite platforms, which has not been answered yet.

Another critical aspect to consider is resilience. A key challenge to a resilient satellite internet system is that it is vulnerable to weather conditions that can disturb the performance and operations of ground components essential for satellite internet, such as satellite antennas used by emerging ground stations and user terminals. These disruptions can significantly affect the uplinks and downlinks essential for satellite internet. This issue is evident in recent studies [21, 12], where weather conditions such as rain and snow are identified to have a significant impact on the performance of the LEO satellite network. For example, rain and snow events can attenuate radiofrequency signals in the Ka/Ku bands used on advanced LEO satellites for ground-space links and cause increased failure rates [21]. General weather forecasts are not sufficient for inferring specific conditions affecting ground components because weather events can lead to complex and varied conditions on these components. A CV-based sensing solution shows great promise for detecting and recognizing the specific conditions of ground components. However, how to design an efficient and generalizable method and apply it to essential ground components, such as satellite antennas, has not been addressed in the literature.

This paper addresses these two critical challenges by leveraging machine learning (ML) techniques. Chapter 2 focuses on the design and evaluation of ML models specifically tailored for satellite object detection, addressing the need for precise and efficient collision assessment and avoidance. Chapter 3 examines the impact of weather-related disruptions on satellite-ground communications, introducing a transfer learning (TL)-based approach to ensure reliable connectivity under adverse conditions. Together, these contributions aim to advance the capabilities of LEO satellite networks, addressing key challenges to their sustainable and resilient operation.

The results presented in this paper have been disseminated through peer-reviewed publications. Chapter 2 is based on work published in [36], and Chapter 3 is based on work published in [35].

Chapter 2

Sensing for Space Safety and Sustainability: A Deep Learning Approach with Vision Transformers

2.1 Background

Traditional sensing technologies for SOD, such as radar and LiDAR sensors, are not efficient for low-Earth orbit (LEO) satellites due to their high power consumption, which can take up to 45% of a satellite's power budget. Vision sensors can generally provide cost-effective solutions with low power consumption. However, their effectiveness has not been fully explored for SOD tasks. The popular object detection models, such as You Only Look Once (YOLO) based models [8, 29, 30], Vision Transformer (ViT) based models [18, 3], focus on the regular detection tasks, and how they perform in SOD tasks is unknown. These models face different limitations when applied to SOD tasks. CNN-based models, such as YOLO, struggle to detect small satellites due to their inability to capture the surrounding context. ViT models are capable of capturing global context but require significant computational resources. When computational resources are limited, the performance of ViT models degrades significantly. In this section, we aim to provide an in-depth discussion on the effectiveness of employing vision sensors with CV capability for SOD tasks based on the state-of-the-art and our proposed deep learning (DL) models: GELAN-ViT, GELAN-ViT-Reduced, GELAN-RepViT-Extended, and GELAN-RepViT. Specifically, our proposed models are based on ViT and the Generalized Efficient Layer Aggregation Network (GELAN) architecture in YOLOv9 [30] and consistently show supe-

rior performance in the SOD tasks. We discuss the strengths of each model and detail how their architecture contributes to improving performance, reducing computational costs, and enhancing precision in detecting small satellites in SOD. Additionally, we showcase the generalizability of our models using the Visual Object Classes (VOC) 2012 dataset.

In the domain of CV, object detection has been an important driving force behind many significant advancements. Convolutional Neural Network (CNN) based detectors are the most widely used in CV, where they excel at capturing the local details. An example of CNN-based detectors is the YOLO series, which is known for its balance between speed and accuracy in real-time applications. YOLOv9 builds upon the GELAN architecture and Programmable Gradient Information (PGI), where GELAN enhances feature aggregation across different layers, and PGI mitigates information loss within the network. As a result, YOLOv9 is expected to excel in complex scenarios where preserving information is key to improving prediction accuracy. YOLOv7 [29] excels in detection speed and accuracy for simpler tasks, but its performance may decline in complex object detection tasks due to its simpler architecture. YOLOv7 is effective in scenarios where high-speed inference is required without significantly sacrificing accuracy.

Despite the strengths of CNN-based models, a study by Jiaxu *et al.* [13] suggests that these models often struggle with small object detection, largely due to their limited ability to capture global dependencies. Small objects often occupy only a few pixels, making it difficult for CNNs to utilize minimal local features for accurate detection. Several approaches have been proposed to mitigate this issue. For example, Feature Pyramid Network (FPN) [15] aims to address the challenge by merging feature maps from different layers of the network, combining low-resolution features from deeper layers with high-resolution features from shallower layers. Other approaches, such as increasing the resolution of feature maps as in [5], have also improved small object detection. However, despite these efforts, CNN-based models still struggle with small object detection due to their natural focus on local features.

ViTs have introduced a new path in CV by capturing global dependencies within images using self-attention mechanisms [6]. Unlike CNNs, which excel at local feature extraction, ViTs are adept at modeling long-range relationships of objects within an image. Models such as YOLOX-ViT [3] and MobileViT [18] attempt to combine the strengths of CNNs and ViTs by integrating transformer layers into CNN architectures to improve both detection accuracy and computational efficiency.

As suggested by Rekavandi *et al.* [24], ViTs have shown success in addressing the limitations of CNNs in small object detection tasks. With their global attention mechanisms, ViTs can capture the surrounding context of objects and model long-range dependencies,

performing well in identifying small objects, even when local features are insufficient. However, ViTs face challenges in terms of computational complexity and their need for large training data. As noted in [6], ViTs also lack the inductive biases that CNNs possess, such as translation invariance and local feature extraction, making them less efficient in capturing fine local details. Given the limitations of both CNNs and ViTs, fusing the two allows the model to leverage the strengths and overcome the weaknesses of both approaches.

Previous approaches to integrating CNNs and ViTs, such as YOLOX-ViT and Mobile-ViT, have limitations. These models do not separate local and global feature extraction processes, but they combine them within the same pathways. This can lead to information interference and loss, as the capacity of each layer is stretched to handle both types of features simultaneously. This issue is particularly problematic in complex detection tasks, where balancing detailed local information with a broader global context is crucial.

In this chapter, we will introduce models that incorporate ViT into the GELAN architecture and address the limitations by separating the CNN and ViT paths, allowing each to specialize in their strengths. This mitigates the risk of information loss and optimizes feature extraction. The following section provides an overview of our system model, explaining how we separate these feature extraction processes and detailing the architecture of our proposed models.

2.2 System Model

2.2.1 Overview

Our work focuses on enhancing the performance of GELAN by addressing the inefficiencies caused by combining local features (F_l) and global features (F_g) within the same pathway. Previous work has shown that when F_l and F_g are processed together, they compete for limited neural resources, decreasing performance. This issue arises from the *information bottleneck* principle, which states that neural networks have a finite capacity for transmitting information [27]. When both feature types are forced to share the same pathway, the network's capacity must be divided between them, limiting the amount of information that can be retained for both local and global features as shown in (2.1):

$$C_{\text{total}} = C(F_l) + C(F_g) \tag{2.1}$$

- $C(F_l)$ represents the capacity allocated to local features,

- $C(F_g)$ represents the capacity allocated to global features.

In this setup, the network may have trouble balancing F_l and F_g, leading to inefficient learning and potential information loss. This bottleneck can limit the model's ability to optimize both feature types. To address this, we separate the local and global feature extraction processes into distinct paths, allowing each to specialize in learning its feature type without interference. This separation ensures that local and global features do not compete for the same neural resources, which increases the model's capacity to process and retain more information as shown in (2.2):

$$C_{\text{local}} = C(F_l), \quad C_{\text{global}} = C(F_g) \tag{2.2}$$

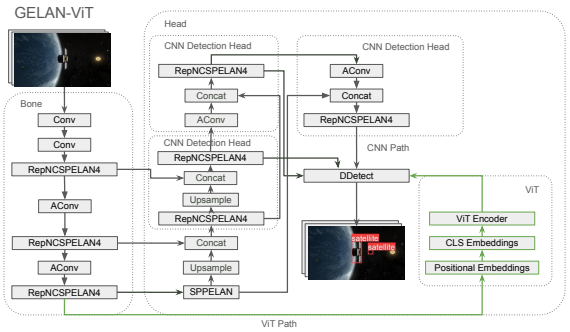
By removing the competition between local and global features, the model can retain and utilize more information from both feature types, allowing for more contextually informed predictions. We hypothesize that this separation increases the model's ability to retain critical information, resulting in better object detection performance.

2.2.2 Model Architecture

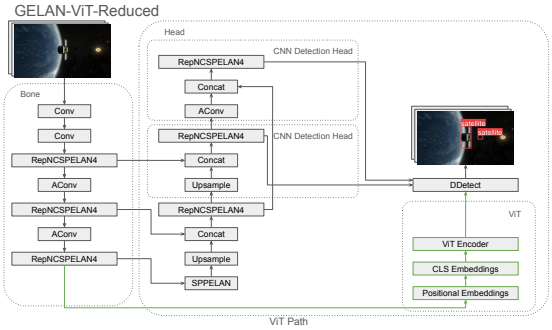
To address the issues caused by combining local and global feature extraction in the same pathway, we propose four novel architectures: GELAN-ViT, GELAN-ViT-Reduced, GELAN-RepViT-Extended, and GELAN-RepViT. All our proposed models follow the GELAN architecture, which comprises multiple modular blocks known as AnyBlock modules. These AnyBlock modules allow for integrating various computational blocks within the network, enhancing its adaptability and performance.

We have based our models on the Reduced-GELAN-s model, a scaled-down version of the original GELAN-s architecture with the number of neurons in each layer reduced to 25% to lower computational complexity. We selected GELAN-s based on our experience and scaled it down to balance computational efficiency and model performance. By reducing the number of neurons, the architecture lowers computational requirements, creating room for additional ViT integration and making the model less memory-intensive while maintaining its core functionality.

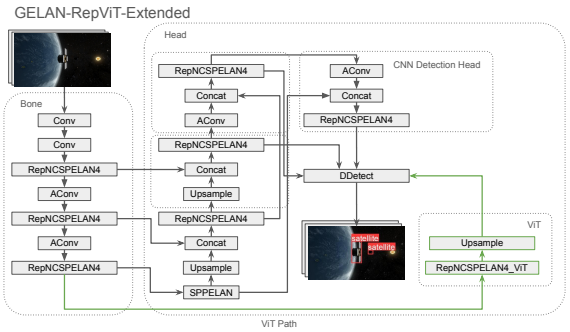
By incorporating ViT into the Reduced-GELAN-s backbone, our proposed models capture global features while retaining the CNN-based architecture for local feature extraction. While ViT effectively captures global features, incorporating it directly into the body of



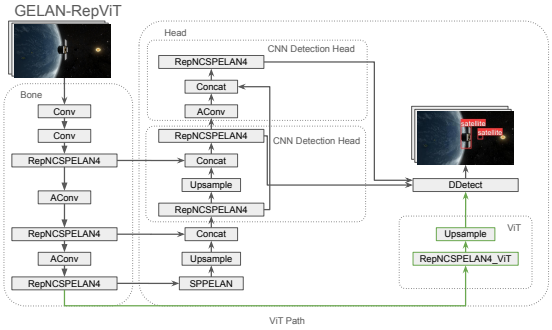
(a) The architecture of the GELAN-ViT model.



(b) The architecture of the GELAN-ViT-Reduced model.



(c) The architecture of the GELAN-RepViT-Extended model.



(d) The architecture of the GELAN-RepViT model.

Figure 2.1: Architectures of the proposed models.

the network would mix global and local feature extraction within the same pathway, potentially leading to information interference. Additionally, ViT's self-attention mechanism is computationally expensive, and integrating it into the body would further increase computational complexity. Through extensive experimentation, we found that placing ViT parallel to the CNN head of the GELAN best balances computational efficiency and detection accuracy.

GELAN-ViT

The GELAN-ViT architecture, shown in Fig. 2.1(a), builds upon the Reduced-GELAN-s model by directly integrating core ViT components, including CLS embeddings, positional encodings, and a transformer encoder. The transformer encoder processes the output from

the last layer of the GELAN-s backbone, leveraging spatial information about object relationships provided by the CLS embeddings and positional encodings. The global features extracted by the ViT path are merged with the CNN-based features within YOLOv9's DDetect layer to enhance detection performance.

To simplify the architecture and improve computational efficiency, we introduce the GELAN-ViT-Reduced model shown in Fig. 2.1(b), where one detection head is removed.

As shown in Fig. 2.1(a) and Fig. 2.1(b), the ViT path runs parallel to the original CNN-based local feature extraction path in both models. The outputs of both paths are combined in the detection layers to produce the final object detection predictions.

GELAN-RepViT-Extended

The GELAN-RepViT-Extended architecture, depicted in Fig. 2.1(c), extends the Reduced-GELAN-s model by adding a branch that integrates RepNCSPELAN4_ViT, an enhanced version of RepNCSPELAN4 augmented with ViT encoder layers, as proposed by Aubard et al. [3].

The GELAN-RepViT architecture, shown in Fig. 2.1(d), is a streamlined version of the GELAN-RepViT-Extended model, with one detection head removed to simplify the architecture.

As shown in Fig. 2.1(c) and Fig. 2.1(d), the RepNCSPELAN4_ViT block operates in parallel with the local feature extraction path. The global features extracted from the ViT branch are passed through additional Upsample layers before merging with the CNN-based features using YOLOv9's DDetect layer.

2.3 Experiments

2.3.1 Overview

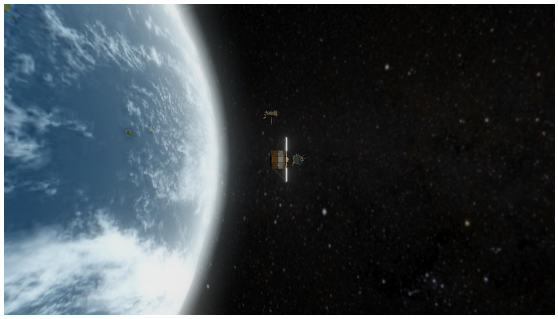
In the experiments, we compared our proposed model with GELAN-t [30], YOLOv9-t [30], YOLOv7-t [29], YOLOX-s [8], YOLOX-ViT-s [3], and MobileViT [18] (with width multiplier of 0.5) using the SOD and VOC 2012 datasets. The experiments were performed on a High-Performance Computing (HPC) cluster with an NVIDIA V100 16 GB GPU for training and testing. We trained each model for 1000 epochs. To measure computational efficiency, we utilized the `seff` command to monitor memory usage and the `thop` library to calculate the GFLOPs and model parameters.

In addition to the HPC environment, we evaluated model performance on a Jetson Nano Orin Developer Kit to understand the real-time applicability of a mini-computer. We used the `tegrastats` tool to track memory usage and the CPU and GPU power consumption, allowing us to capture resource usage continuously throughout each run. We collected these logs during each inference session, calculating both the mean and peak memory usage for each run. Then, we averaged these mean and peak values across multiple runs to generate the final results, resulting in a representative metric and its corresponding confidence interval for each model tested.

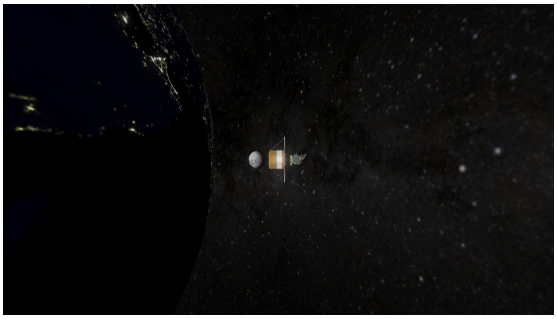
We recorded baseline memory usage before running any model inference and subtracted this baseline from the mean and peak usage captured during each run. This adjustment excluded memory consumption attributable to the operating system and background processes, ensuring that the reported metrics reflect only the memory usage and power consumption specifically required by each model.

The results presented in this section are averaged over 25 consecutive runs. The first five runs were discarded to minimize the impact of resource allocation times, leaving the average calculated from the remaining 20 runs. The experiments are conducted with an image resolution of 640 px $\times$ 640 px for the SOD Dataset (SODD) and 320 px $\times$ 320 px for the VOC 2012 dataset. The SODD is tailored for SOD tasks involving LEO satellites, focusing on satellite object classes and celestial bodies. In contrast, the VOC 2012 dataset serves as a well-established benchmark for general object detection tasks, containing 20 distinct object classes. The inference time, memory usage, and power consumption are collected using the batch size of one to simulate real-time object detection scenarios where images are processed individually.

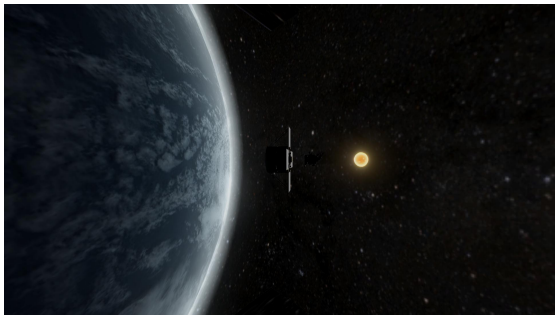
The results in Table 1 provide a comparative analysis of our proposed models, including GELAN-ViT, GELAN-ViT-Reduced, GELAN-RepViT-Extended, and GELAN-RepViT, against state-of-the-art object detection models across the SOD and VOC 2012 datasets. Our models demonstrate competitive performance with leading models in terms of mean Average Precision (mAP), specifically mAP at a 50% IoU threshold (mAP50) and mAP across a range of 50% to 95% IoU (mAP50:95). All proposed models show a marked reduction in computational complexity, measured in GFLOPs. Among these, GELAN-ViT achieves the best results on the SODD, while GELAN-RepViT demonstrates superior performance on the VOC 2012 dataset.



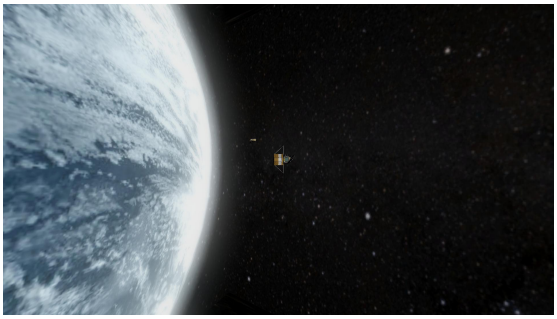
(a) Earth is visible in a bright environment, with two satellites visible.



(b) The Moon is visible in a dark environment with Earth partially illuminated.



(c) The Sun is visible, introducing brightness into the scene, with Earth also visible.



(d) Two satellites in view, one being a small object blending into the starry background.

Figure 2.2: Sample images from the SODD demonstrating differences in lighting conditions and visibility of celestial bodies such as the Earth, Moon, and Sun.

2.3.2 Dataset

The SODD is specified for satellite-based object detection tasks. It includes images from three satellites, each providing diverse perspectives and environmental conditions, including celestial bodies of the Moon, Sun, and Earth. Fig. 2.2(a) is an example where we can observe Earth with two satellites in view. Fig. 2.2(b) captures the Moon in a dark environment, with Earth and the satellite illuminated against a dim background. Meanwhile, Fig. 2.2(c) shows the Sun visible, with Earth and the satellite in a relatively dimmer, less bright background. Fig. 2.2(d) shows Earth with two satellites in view, one of which is a small object blending into the starry background, highlighting the challenge of small object detection in SOD.

The VOC 2012 dataset, on the other hand, is widely used for standard object detection tasks in CV. It contains images of everyday objects captured from ground-level

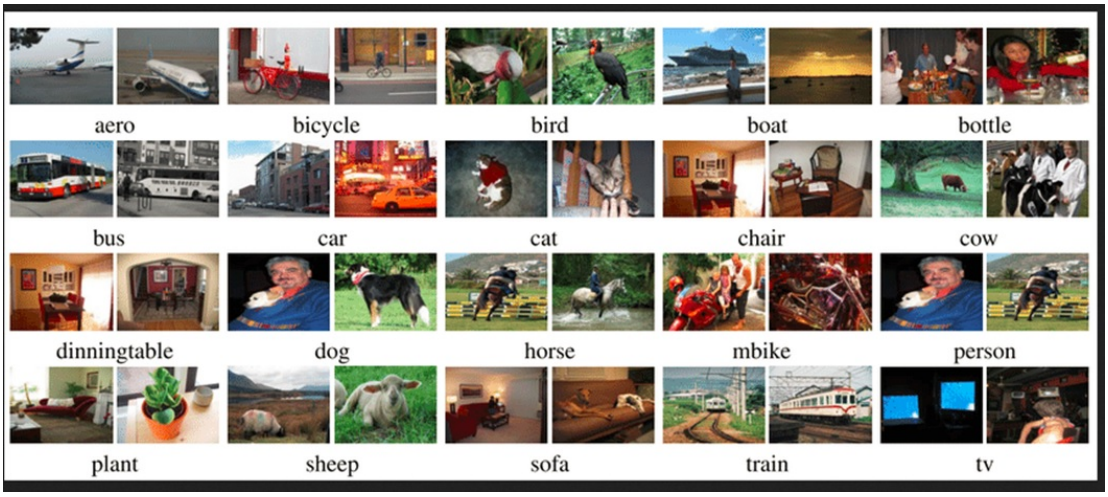


Figure 2.3: Sample images from the VOC 2012 dataset featuring 20 object classes such as animals, vehicles, and household items.

perspectives. The dataset features a diverse range of 20 object classes, such as animals, vehicles and people, as shown in Fig. 2.3. The VOC dataset is more complex compared to the SODD, as it presents a greater range of object types, environments, and visual characteristics that challenge how detection models handle effectively.

2.3.3 Evaluation Metrics

Our evaluation strategy focuses on key metrics in object detection tasks with regard to Intersection over Union (IoU), specifically mAP50 and mAP50:95. The metrics we use are defined below.

- **Average Precision at a i% IoU threshold (AP_i):** Average Precision at IoU threshold i is calculated as the area under the precision-recall curve for IoU threshold i ($p_i(r)$).

$$AP_i = \int_0^1 p_i(r)\, dr$$

- **mAP50:** This is the average of AP_{50} across all the classes C:

$$mAP50 = \frac{1}{C} \sum_{i=1}^{C} AP_{50,i}$$

Table 2.1: Comparison of different models on SODD and VOC 2012 datasets.

Model	Dataset	mAP50	mAP50:95	GFLOPs	Parameters
GELAN-t [30]	SODD	0.944	0.802	7.3	1,878,819
YOLOv9-t [30]	SODD	**0.951**	**0.830**	10.7	2,659,286
GELAN-ViT	SODD	0.947	0.815	**5.7**	7,327,012
GELAN-ViT-Reduced	SODD	**0.952**	**0.818**	**5.4**	7,035,043
GELAN-RepViT	SODD	0.950	**0.836**	**5.2**	**1,247,811**
GELAN-RepViT-Extended	SODD	0.944	**0.818**	**5.4**	**1,559,644**
YOLOv7-t [29]	SODD	**0.956**	0.759	13.0	6,007,596
YOLOX-s [8]	SODD	0.858	0.584	15.2	5,032,866
YOLOX-ViT-s [3]	SODD	0.847	0.575	15.1	4,884,834
MobileViT [18]	SODD	0.624	0.449	6.3	**1,189,706**
GELAN-t [30]	VOC 2012	0.605	0.440	7.4	1,917,148
YOLOv9-t [30]	VOC 2012	**0.606**	**0.446**	11.1	2,666,696
GELAN-ViT	VOC 2012	**0.619**	**0.460**	5.9	7,350,024
GELAN-ViT-Reduced	VOC 2012	**0.607**	0.441	**5.7**	7,053,620
GELAN-RepViT	VOC 2012	0.602	**0.439**	**5.2**	**1,268,180**
GELAN-RepViT-Extended	VOC 2012	0.604	0.435	**5.4**	**1,564,568**
YOLOv7-t [29]	VOC 2012	0.589	0.390	13.3	6,066,402
YOLOX-s [8]	VOC 2012	0.585	0.351	15.3	5,038,395
YOLOX-ViT-s [3]	VOC 2012	0.565	0.340	15.1	4,890,363
MobileViT [18]	VOC 2012	0.535	0.374	6.3	**1,194,589**

where $AP_{50,i}$ represents the AP_{50} for class i.

- **mAP50:95:** This value is averaged across all the classes C:

$$mAP50:95 = \frac{1}{C}\sum_{i=1}^{C}\left(\frac{1}{10}\sum_{j=1}^{10}AP_{50+5(j-1),i}\right)$$

2.3.4 Hyperparameters

Some hyperparameters were adjusted for the training of the evaluated models. These adjustments were made manually to achieve faster convergence and obtain meaningful results within 1000 epochs without advanced hyperparameter optimization techniques.

- **Learning Rate (Initial, Final):** The initial learning rate was set at 0.01, with a final learning rate of 0.1 using the OneCycleLR policy.

- **Loss Gains:**
  - **Box Loss Gain:** Set to 0.05, this parameter controls the weight of the bounding box regression loss.
  - **Classification Loss Gain:** Set to 0.5, this parameter controls the importance placed on the accuracy of classifying objects within the bounding boxes.
  - **Objectness Loss Gain:** Set to 1.0, this parameter controls the emphasis on detecting the presence of objects within predicted bounding boxes.
- **Data Augmentation:**
  - **HSV-Saturation:** Set to 0.7, this parameter controls the variation in color saturation within training images.
  - **HSV-Value:** Set to 0.4, this parameter controls the brightness of images.
  - **Scale Augmentation:** Set to 0.5, this parameter controls the scaling of images during training.
  - **Mixup:** Disabled (set to 0.00), this parameter controls the blending of multiple images during training, which was turned off to avoid introducing unnecessary complexity.
- **Anchor-Multiple Threshold:** Set to 4.0, this threshold determines the conditions under which anchor boxes are selected.

Other hyperparameters were kept at the model's default values from the model repository. These defaults were selected based on standard practices.

2.3.5 Performance on SODD

The performance of models in the SODD is summarized in Table 2.1. Our proposed models, GELAN-ViT, GELAN-ViT-Reduced, GELAN-RepViT-Extended, and GELAN-RepViT, are compared with GELAN-t, YOLOv9-t, YOLOv7-t, YOLOX-s, YOLOX-ViT-s, and MobileViT. The metrics used for the evaluation include mAP50 at 1000 epochs, mAP50:95 at 1000 epochs, GFLOPs, and the number of parameters. In calculating GFLOPs, YOLOv7-t and YOLOv9-t default to an input size of 640 px $\times$ 640 px, while MobileViT, YOLOX-s, and YOLOX-ViT-s use their respective input sizes. To ensure consistency in our comparison, we fixed the GFLOPs calculation to an input size of 640 px $\times$ 640 px for all models.

GELAN-ViT achieves an mAP50 of 0.947 at 1000 epochs, while GELAN-ViT-Reduced slightly outperforms with an mAP50 of 0.952. Additionally, GELAN-RepViT-Extended records an mAP50 of 0.944, and GELAN-RepViT reaches an mAP50 of 0.950. All four models show a difference of less than 0.02 in mAP50 compared to GELAN-t with 0.944, YOLOv7-t with 0.956, and YOLOv9-t with 0.951. Models with significantly lower mAP50 include YOLOX-s with 0.858, YOLOX-ViT-s with 0.847, and MobileViT with 0.624.

Regarding mAP50:95, GELAN-ViT achieves 0.815 at 1000 epochs, while GELAN-ViT-Reduced and GELAN-RepViT-Extended achieves 0.818. GELAN-RepViT further improves to 0.836. In comparison, GELAN-t achieves an mAP50:95 of 0.802, and YOLOv9-t achieves an mAP50:95 of 0.830. Models with significantly lower mAP50:95 scores include YOLOv7-t with 0.759, YOLOX-s with 0.584, YOLOX-ViT-s with 0.575, and MobileViT with 0.449. Although YOLOv7-t achieves a high mAP50, it performs significantly lower in mAP50:95. YOLOX-s, YOLOX-ViT-s, and MobileViT maintain lower performance compared to YOLOv9-t and GELAN-t.

All models deliver comparable performance to YOLOv9-t and GELAN-t while maintaining lower GFLOPs. Compared to GELAN-t, GELAN-ViT reduces GFLOPs by 1.6, GELAN-ViT-Reduced by 2.0, and GELAN-RepViT-Extended by 1.9, while GELAN-RepViT reduces GFLOPs by 2.1.

2.3.6 Performance on VOC 2012 Dataset

The performance of our models in the VOC 2012 dataset is also summarized in Table 2.1. The metrics used for evaluation include mAP50 at 1000 epochs, mAP50:95 at 1000 epochs, GFLOPs, and the number of parameters. The input size for the VOC dataset is standardized to 320 px $\times$ 320 px. However, for consistency in GFLOPs calculation across all models, we fixed the GFLOPs calculation to an input size of 640 px $\times$ 640 px, regardless of the original input size used in training.

In the VOC 2012 dataset, GELAN-ViT achieves the highest mAP50 of 0.619 at 1000 epochs, followed by GELAN-ViT-Reduced with 0.607, and GELAN-RepViT-Extended at 0.604. GELAN-RepViT delivers an mAP50 of 0.602, slightly lower than its reduced counterpart. GELAN-ViT and GELAN-ViT-Reduced outperform GELAN-t's 0.605 and YOLOv9-t's 0.606, while GELAN-RepViT-Extended and GELAN-RepViT underperform.

GELAN-ViT represents a 0.014 increase over GELAN-t and a 0.013 increase over YOLOv9-t. GELAN-ViT-Reduced shows a 0.002 increase compared to GELAN-t and a 0.001 increase compared to YOLOv9-t. GELAN-RepViT-Extended shows a 0.001 decrease compared to GELAN-t and a 0.002 decrease compared to YOLOv9-t, while GELAN-

RepViT shows a 0.003 decrease compared to GELAN-t and a 0.004 decrease compared to YOLOv9-t. Models with lower mAP50 include YOLOX-s with 0.585, YOLOX-ViT-s with 0.565, and MobileViT with 0.535.

Regarding mAP50:95, GELAN-ViT leads with 0.460, while GELAN-ViT-Reduced achieves 0.441, GELAN-RepViT achieves 0.439, and GELAN-RepViT-Extended achieves 0.435. In comparison, GELAN-t records an mAP50:95 of 0.440, and YOLOv9-t achieves 0.446. GELAN-ViT outperforms GELAN-t by 0.02 and YOLOv9-t by 0.014. GELAN-ViT-Reduced's mAP50:95 is 0.001 higher than GELAN-t but 0.005 lower than YOLOv9-t. GELAN-RepViT's mAP50:95 are 0.001 lower than GELAN-t and 0.007 lower than YOLOv9-t. GELAN-RepViT-Extended, while underperforming the reduced models, is 0.005 lower than GELAN-t and 0.011 lower than YOLOv9-t. Models with lower mAP50:95 scores include YOLOX-s with 0.351, YOLOX-ViT-s with 0.340, and MobileViT with 0.374.

While GELAN-ViT achieves superior performance and the other three proposed models deliver performance comparable to YOLOv9-t and GELAN-t, all proposed models maintain low GFLOPs. Compared to GELAN-t, GELAN-ViT reduces GFLOPs by 1.5, GELAN-ViT-Reduced by 1.7, GELAN-RepViT by 2.2, and GELAN-RepViT-Extended reduces GFLOPs by 1.9.

2.3.7 Model Stability

In the SODD, YOLOv7-t achieves the mAP50 in the first place of 0.956, surpassing YOLOv9-t, which scores 0.951. However, its mAP50:95 is significantly lower at 0.759 compared to YOLOv9-t's 0.830. This indicates that while YOLOv7-t shows strong performance in mAP50, it exhibits inconsistent precision across different IoU thresholds and struggles to achieve higher accuracy at these more demanding levels.

Although YOLOv7-t's high mAP50 in the simpler SODD highlights its capability in less complex scenarios, this advantage diminishes in the more complex VOC 2012 dataset. In the VOC 2012 dataset, YOLOv7-t's performance is lower than that of YOLOv9-t, with an mAP50 of 0.589 compared to YOLOv9-t's 0.606 and an mAP50:95 of 0.390 compared to YOLOv9-t's 0.446.

GELAN-based models perform better in complex scenarios and demonstrate more stable and consistent performance across various IoU thresholds in both datasets, particularly maintaining higher mAP50:95 values.

GELAN-ViT and GELAN-ViT-Reduced are expected to excel on complex datasets due to their ability to capture detailed patterns. However, their complexity also increases

the risk of overfitting on simpler datasets. As demonstrated in the experiments, GELAN-ViT performs exceptionally well on the VOC 2012 dataset, achieving the highest mAP50 of 0.619 and mAP50:95 of 0.460, making it ideal for complex object detection tasks. GELAN-ViT-Reduced was designed to reduce complexity and maintain competitive performance on VOC 2012. It also offers an advantage on the simpler SODD, where it surpasses GELAN-ViT with a higher mAP50 of 0.952 and mAP50:95 of 0.818.

GELAN-RepViT-Extended, with the advantage of its streamlined architecture, reduces computational cost while maintaining a similar performance comparable to GELAN-ViT-Reduced on the SODD. GELAN-RepViT further simplifies the architecture, allowing it to outperform GELAN-RepViT-Extended with higher mAP50 and mAP50:95 on the SODD. On VOC 2012, both GELAN-RepViT-Extended and GELAN-RepViT achieved similar results. GELAN-RepViT records a higher mAP50:95 of 0.439, though its mAP50 of 0.602 is slightly lower than the 0.604 achieved by GELAN-RepViT-Extended.

Table 2.2: Comparison of Inference Time and Memory Usage for different models on V100 GPU

Model	**Average Inference Time (ms)**	**Memory Usage (GB)**
GELAN-t	17.38	1.82
YOLOv9-t	21.69	1.63
GELAN-ViT	17.78	1.50
GELAN-ViT-Reduced	16.49	1.48
GELAN-RepViT-Extended	14.92	1.44
GELAN-RepViT	10.50	1.43
YOLOv7-t	4.99	6.30
YOLOX-s	14.61	2.90
YOLOX-ViT-s	15.09	3.50
MobileViT	9.03	1.51

2.3.8 Memory Usage

Memory usage results are summarized in Table 2.2, with values recorded using a 640 px $\times$ 640 px input size, averaged over three runs. GELAN-RepViT consumed 1.43 GB of memory, demonstrating superior memory efficiency. GELAN-RepViT-Extended showed slightly higher usage at 1.44 GB, while GELAN-ViT-Reduced and GELAN-ViT consumed 1.48 GB and 1.50 GB, respectively.

In comparison, GELAN-t required 1.82 GB of memory, while YOLOv9-t consumed 1.63 GB. YOLOv7-t had the highest memory usage, at 6.30 GB. YOLOX-s and YOLOX-ViT-s used 2.9 GB and 3.5 GB of memory, respectively, while MobileViT consumed the least at 1.51 GB, which aligns with its smaller model size and lower GFLOPs count.

The relatively high memory consumption of YOLOv7-t is consistent with their superior performance and greater GFLOPs demands. In contrast, our proposed models demonstrated superior memory efficiency compared to GELAN-t, YOLOX-s, and YOLOX-ViT-s. The reduction in memory usage for our models is largely due to the scaling of the Reduced-GELAN-s backbone.

Although GELAN-ViT and GELAN-ViT-Reduced have high parameter counts, this does not translate into higher memory usage. The increased parameters are primarily associated with the ViT module, which has a quadratic relationship with input length, as described in [28]. Since our model processes the input through the backbone before feeding it into the ViT path, we effectively reduce ViT's input length, leading to memory efficiency.

2.3.9 Inference Time

GELAN-RepViT-Extended recorded an inference time of 14.92 ms, with GELAN-RepViT improving further to 10.50 ms. Simplifying the CNN layers in GELAN-RepViT resulted in a more substantial reduction, with a 4.42 ms decrease compared to the GELAN-RepViT.

Conversely, GELAN-ViT showed an inference time of 17.78 ms, while GELAN-ViT-Reduced achieved a slightly better time of 16.49 ms. GELAN-ViT-Reduced only marginally improved inference time by 1.29 ms compared to GELAN-ViT, the reduction in CNN layers did not significantly decrease overall time. This suggests that in the GELAN-ViT models, the ViT path significantly contributes to the overall inference time, and reducing CNN layers alone has a limited impact on performance.

As expected, YOLOv9-t had the highest inference time at 21.69 ms due to its larger model size. GELAN-t posted an inference time of 17.38 ms, comparable to GELAN-ViT and GELAN-ViT-Reduced but higher than GELAN-RepViT-Extended and GELAN-RepViT. The anchor-free design of YOLOX-s and YOLOX-ViT-s enabled inference times of 14.61 ms and 15.09 ms, respectively. MobileViT, with its smaller model size, achieved an inference time of 9.03 ms. YOLOv7-t demonstrated the fastest inference time of 4.99 ms among the compared models, aligning with its design to prioritize speed. However, this speed comes at the expense of significantly higher memory usage and GFLOPs.

Table 2.3: Comparison of Inference Time and Memory for different models on Jetson Orin Nano

Model Name	Average Inference Time (ms)	Peak RAM (GB)	Average RAM
GELAN-t	47.22	2.26	2.25
YOLOv9-t	63.16	2.24	2.22
GELAN-ViT	56.33	2.46	2.45
GELAN-ViT-Reduced	52.40	2.34	2.34
GELAN-RepViT-Extended	44.30	2.21	2.21
GELAN-RepViT	36.72	2.37	2.35
YOLOv7-t	21.92	3.52	3.46
YOLOX-s	44.92	2.45	2.42
YOLOX-ViT-s	43.38	2.41	2.38
MobileViT	30.71	2.26	2.24

Table 2.4: Comparison of and Power Consumption for different models on Jetson Orin Nano

Model Name	Peak Power Consumption	Average Power Consumption
GELAN-t	1599.5 mW	1363.8 mW
YOLOv9-t	1714.8 mW	1472.7 mW
GELAN-ViT	1527.5 mW	1166.5 mW
GELAN-ViT-Reduced	1662.3 mW	1433.9 mW
GELAN-RepViT-Extended	1605.7 mW	1409.4 mW
GELAN-RepViT	1594.7 mW	1277.6 mW
YOLOv7-t	3785.3 mW	2940.3 mW
YOLOX-s	3452.7 mW	2708.6 mW
YOLOX-ViT-s	3659.8 mW	2731.5 mW
MobileViT	1951.8 mW	1678.4 mW

2.3.10 Inference Performance on the Jetson Orin Nano

Table 2.3 and 2.4 summarized the inference times, memory usage, and power consumption recorded on the Jetson Orin Nano. Fig. 2.4 visualizes the memory usage and power consumption in the bar charts, with 95% Confidence Level (CL) provided. Among the models, YOLOv9-t demonstrates the highest inference time at 63.16 ms. GELAN-ViT records the second-highest inference time of 56.33 ms, whereas GELAN-RepViT-Extended achieves a lower inference time of 44.30 ms. YOLOX-s and YOLOX-ViT-s achieve inference times of 44.92 ms and 43.38 ms, respectively. MobileViT, with an inference time of 30.71 ms, highlights the efficiency of a low-GFLOPs model. YOLOv7-t achieves the fastest inference

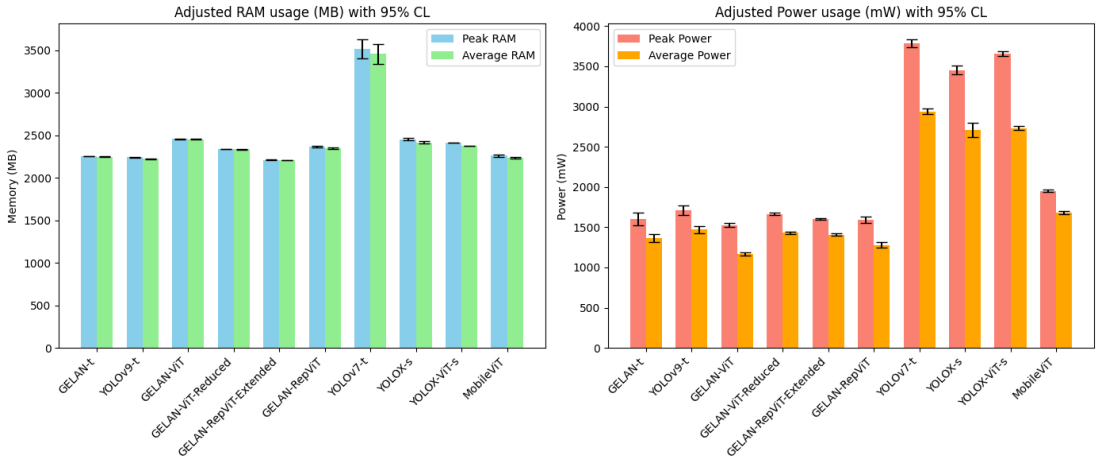


Figure 2.4: Bar charts of RAM usage and power consumption, with the 95% CL indicated for each model

time of 21.92 ms. However, this speed improvement is accompanied by higher memory and power requirements, presenting a trade-off that may not be suitable for deployment in resource-constrained environments.

In terms of the average power consumption, YOLOv9-t has an average power consumption of 1472.7 mW, while GELAN-t, GELAN-ViT, GELAN-ViT-Reduced, GELAN-RepViT-Extended, and GELAN-RepViT consume 1363.8 mW, 1166.5 mW, 1433.9 mW, 1409.4 mW, and 1277.6 mW, respectively. MobileViT shows a slightly higher power consumption of 1678.4 mW. YOLOv7-t prioritizes speed over power efficiency, consuming the highest amount of power at 2940.3 mW, followed by YOLOX-s at 2708.6 mW and YOLOX-ViT-s at 2731.5 mW. These higher power demands suggest that while these models excel at high-speed inference, they may be less suited to power-sensitive applications. In contrast, GELAN-RepViT and MobileViT, with their lower power consumption, offer a preferable power consumption in resource-constrained environments.

In the pattern for peak power consumption, YOLOv7-t, YOLOX-s, and YOLOX-ViT-s show the highest peak power consumption at 3785.3 mW, 3452.7 mW, and 3659.8 mW, respectively. The GELAN models exhibit peak power consumption ranging from 1527.5 mW for GELAN-ViT to 1662.3 mW for GELAN-ViT-Reduced, indicating a stable power profile across the GELAN series. MobileViT peaks at 1951.8 mW, though slightly higher than other GELAN-based models, remaining within an efficient range.

2.3.11 Discussion of Model Superiority

ViTs utilize the self-attention mechanism to learn global features by capturing long-range dependencies within an image, which is crucial for complex object detection tasks requiring scene understanding and identifying object interactions. ViTs are particularly effective in detecting small objects, as they can leverage surrounding contextual information, even when local details are sparse, as highlighted by Rekavandi *et al.* [24].

On the other hand, CNNs excel at extracting detailed features through local patterns. They are particularly effective in recognizing edges, textures, and shapes, essential for accurately identifying individual objects within an image. By refining these local details, CNNs construct precise representations of objects, ensuring that their specific characteristics are captured effectively.

Our model optimizes feature extraction by separating the ViT and CNN pathways. This allows CNNs to specialize in local feature extraction, while ViTs focus on capturing the global context. By separating these paths, our model prevents interference between the local and global feature learning processes, allowing each network to specialize without distraction. This design also addresses the information bottleneck problem: when local and global features share the same pathway, information is lost due to the limited capacity of the neurons. By using separate pathways, each type of feature is retained more effectively. Later in the network, these features are fused, combining local and global strengths to improve overall performance.

In comparison, YOLOX-ViT-s's performance decreases due to the competition between local and global feature extraction within the same pathway. This shared processing introduces an information bottleneck, limiting the model's ability to capture both feature types effectively. A similar observation was made by Aubard *et al.* [3], where integrating ViT increased false positives and contributed to potential overfitting. These findings highlight a trade-off between the benefits of enhanced global feature extraction and the efficiency required for accurate object detection, as reflected in YOLOX-ViT-s's performance.

A similar pattern is observed in MobileViT, but since its architecture is designed to be lightweight with fewer parameters and narrower layers, the impact of the information bottleneck becomes even more noticeable. MobileViT's reduced capacity exacerbates the challenge of balancing F_l and F_g, leading to greater information loss and a more significant decline in accuracy compared to larger models of YOLOX-ViT-s or GELAN-ViT.

Chapter 3

An Efficient and Generalizable Transfer Learning Method for Weather Condition Detection on Ground Terminals

3.1 Background

In general, object detection based on CNN can be classified into one-stage and two-stage detectors. Two-stage detectors, such as R-CNN [9], prioritize detection accuracy by identifying regions of interest and then classifying and refining the bounding boxes. In contrast, one-stage detectors, such as YOLO series [29] and Single-Shot Detection (SSD) [17], streamline this process by directly predicting bounding boxes and class confidences. One-stage detectors face challenges in detecting small objects but offer faster training times and require fewer training images [13, 17]. R-YOLO, proposed by Wang *et al.* [32], has specifically enhanced YOLO for adverse weather.

Despite these developments, a common challenge for object detection models lies in the reliance on large datasets and the potential inclusion of irrelevant features that can degrade performance in specific applications. Moreover, many models are developed with minimal integration of domain-specific human knowledge, which could otherwise enhance their accuracy without increasing complexity. This gap is particularly problematic in scenarios with limited data availability, where incorporating human knowledge could help

mitigate the impact of scarce or noisy datasets while tailoring models to the problem's unique requirements.

TL is an ML technique that adapts knowledge learned from one task to another related task. The study [34] highlights that the TL method is particularly advantageous when labeled data is scarce or specific conditions are underrepresented in domains.

To the best of our knowledge, our work first provides a generalizable vision-based solution to weather condition detection for ground terminals. By applying TL, we address the challenge of data scarcity by adapting a model to classify the conditions of objects across varied weather conditions accurately. Feature-based TL specifically focuses on manipulating the data features extracted by the pre-trained model to better align with the specifics of the target application.

In this part of the paper, we focus on designing an effective TL method for detecting weather-related conditions on satellite antennas. The You Only Look At CoefficienTs (YOLACT) framework is used to apply model transfer for segmentation and feature isolation. We will simulate an environment with limited data and address the challenge of data scarcity by adapting our model to classify the conditions of satellite antennas in various weather conditions. By using the extracted features from a constrained dataset, we have developed a model that exceeds the detection accuracy of existing methods under similar limitations.

We then explore the multi-class classification of satellite antennas on ground terminals under various weather conditions. Specifically, we analyze conditions in two scenarios: the initial scenario where we classify antennas as either snow-covered or normal conditions, and the extended scenario that classifies antennas under snow, wet, and normal conditions. In the initial scenario with 80 training images, our proposed model achieves an accuracy of 88.33% within 50 epochs, outperforming mainstream DL models including the top-performing R-YOLO and Faster R-CNN [25] models, whose accuracy values are 74.16% and 80.00%, after 500 epochs. In the extended scenario with 180 training images, our model reaches an accuracy of 88.33% in 50 epochs, compared to R-YOLO and Faster R-CNN, whose accuracy is 72.22% and 81.11% after 500 epochs, respectively.

While segmentation of satellite antennas using traditional YOLACT techniques is an initial step in our method, this section focuses on the classification processes. Since the segmentation step follows standard YOLACT procedures, we have chosen to concentrate on the novel classification tasks in our study.

3.2 System Model

3.2.1 Overview

Our method utilizes feature-based TL to leverage the knowledge of a model trained on a weather database (source domain, D_S) and apply it to the satellite antenna classification task (target domain,D_T). We first consider the neural network as a function $F(I;W)$ with input I and weights W. The function F is initially trained on D_S to minimize the overall loss on D_S, given by $\sum_{(input,label)\in D_S} L(F(input,W),label)$, where D_S is the source domain, *input* is the input image from D_S, *label* is the corresponding label, and L is the loss function of F.

The goal of feature-based TL is to learn a weight OPT_T that minimizes the loss $L(D_T, OPT_T)$. Given the optimal weight for the source domain OPT_S, we initialize $W = OPT_S$ and train W with data from the target domain D_T to minimize the loss $L(D_T, W)$. Let $MMD(D_S, D_T)$ denote the Maximum Mean Discrepancy [38] between the characteristic distributions of D_S and D_T. As $MMD(D_S, D_T)$ decreases, the model's ability to generalize from D_S to D_T increases.

3.2.2 Pre-Processing: Background Removal Using YOLACT

Background can introduce unnecessary complexity in D_T, increasing the risk of overfitting and degrading model performance, as discussed by [26]. To address this, we have fine-tuned YOLACT [4] specifically for the detection and segmentation of satellite dish antennas (collectively referred to as satellite dishes). We use YOLACT to first apply segmentation masks to the entire image, then filter out the Object of Interest (O_I) using the mask.

The application of YOLACT can be represented as a function Y, transforming D_T into a D_{ST}, therefore focusing on O_I and aiming to minimize MMD between D_S and D_{ST}. This step can be represented as $Y : D_T \rightarrow D_{ST}$, and $MMD(D_S, D_T) \geq MMD(D_S, D_{ST})$. Loss for YOLACT ($L_{YOLACT}$) are combined of three primary losses: classification loss (L_{cls}), box regression loss (L_{box}), and mask loss (L_{mask}). The weights for these losses are 1, 1.5, and 6.125, respectively.

$$L_{YOLACT} = L_{cls} + 1.5 \cdot L_{box} + 6.125 \cdot L_{mask} \tag{3.1}$$

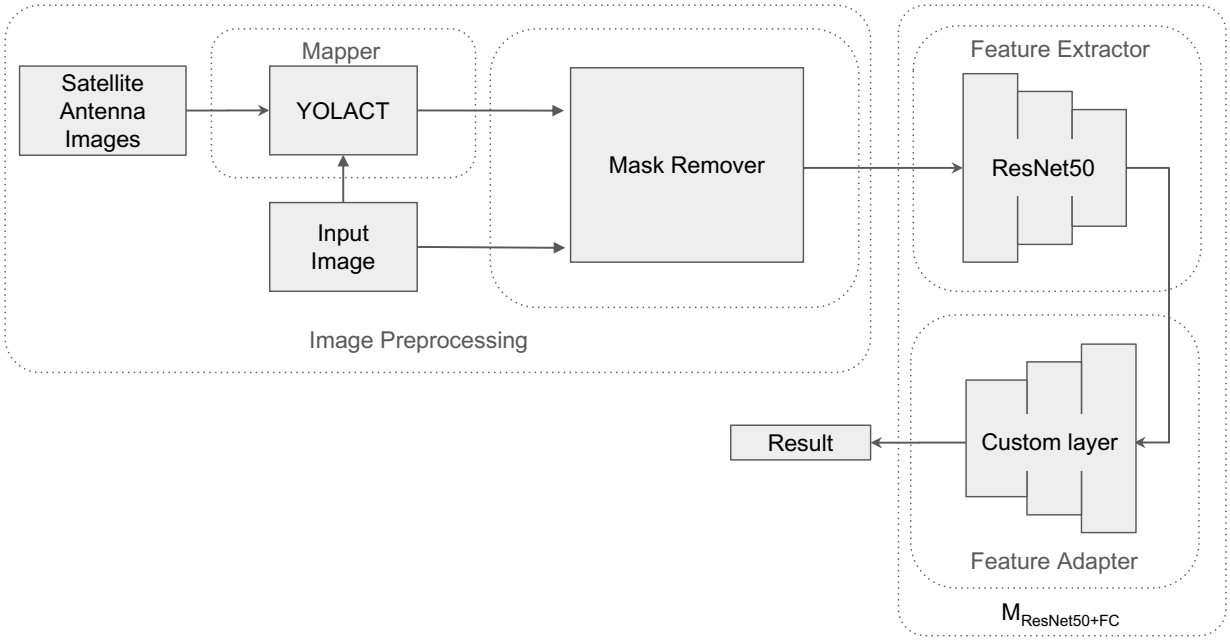


Figure 3.1: The process and architecture of the proposed transfer learning method.

3.2.3 Proposed TL Method

Our proposed method is shown in Fig. 3.1, beginning with fine-tuning YOLACT using satellite antenna images to generate binary masks of satellite dishes in the images. After the fine-tuning stage, YOLACT will take input images and generate a mask. Masks and input images are then processed through Mask Remover to isolate satellite dishes in the image. The processed images are passed through a feature extractor, which extracts relevant features. These extracted features are used by the feature adapter to classify satellite dishes based on weather conditions. The higher layers of ResNet50 are task-specific and should be adjusted with a fine-tuned learning rate to effectively specialize in our task. We have used partial freezing and learning rate adjustment to preserve the generic features learning from D_S for the pre-trained model while allowing the model to adapt to the D_{ST}.

Let $W = W_1, W_2, ..., W_{nl}$ represent the set of weights in the model, where W_i represent the weight of the layer i, and nl represent the total layer in the model. The updates to W_i during the training process on D_{ST} can be represented by $\delta W_i = \alpha \times \nabla L(D_{ST}; W)$ with learning rate α, and gradient $\nabla L(D_{ST}; W)$. During the fine-tuning process, we have set

the α for $W_1, W_2, ..., W_{nl-1}$ to zero, and downscale α to a lower value α', So

$$\delta W_i = \begin{cases} \alpha' \times \nabla L(D_{ST}; W), & \text{for } i = nl \\ 0, & \text{otherwise} \end{cases}$$

3.2.4 Model Architecture

Our model, referred to as $M_{ResNet50+FC}$, is based on pre-trained ResNet50. The last layer of ResNet50 is removed to use the model as a feature extractor, and the output features from the convolutions base are fed in the custom Fully Connected (FC) layers. The choice of ResNet50 is pivotal due to its deep residual learning framework, as [11] demonstrates, addressing the problem of the vanishing gradient. The study [2] also demonstrates the effectiveness of ResNet50 in categorizing weather conditions. We have added two FC layers to learn the features extracted from ResNet50. The first FC layer contains 128 units followed by a dropout layer to prevent overfitting. The second FC layer matches the number of classes in the multi-class classification tasks.

Our model's loss function is cross-entropy loss, defined as

$$L(y, \hat{y}) = -[ylog(\hat{y}) + (1 - y)log(1 - \hat{y})] \tag{3.2}$$

After combining the loss in the procedures of the TL, the final loss is given as

$$L_{overall} = L(y, \hat{y}) + L_{YOLACT}, \tag{3.3}$$

where L_{YOLACT} is the segmentation loss and $L(y, \hat{y})$ is the classification loss.

3.3 Data Preparation

As shown in Fig. 3.2, we pre-processed our data to enhance the model's capability to identify satellite dishes in various weather conditions. We start with initial data preparation, where we collect images of satellite dishes under various conditions. To augment this dataset and enhance its diversity, we utilize two external datasets: the Weather Analysis Dataset [1] which provides images under typical weather conditions such as sunny and cloudy, and the Snow Pile Classification Dataset [20] which supplies images under snow conditions. For the extended scenario which includes wet conditions, we incorporate an additional dataset [22] that offers images in rain conditions, and we use DALL-E to generate images of satellite dishes in wet conditions due to the scarcity of available data.

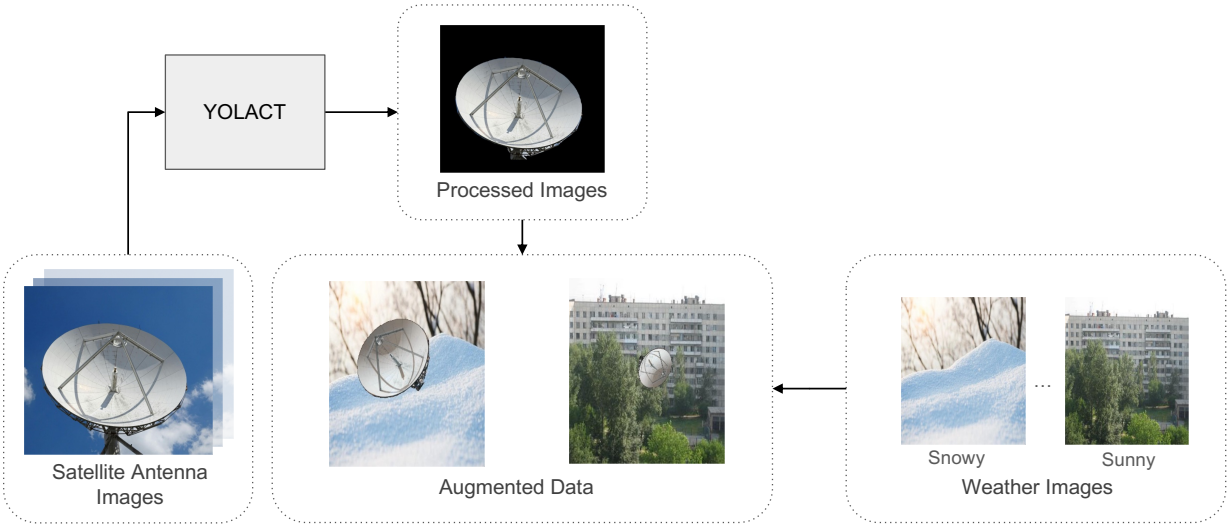


Figure 3.2: A process of data generation. Satellite dishes from original images are segmented using YOLACT and combined with various weather backgrounds to generate diverse training images.

We then move to background removal, employing YOLACT to create binary masks. In this process, pixels corresponding to the object are assigned one binary value, distinguishing them as the O_I. After that, we use the binary mask to isolate the satellite dishes from their environments.

Finally, we perform image enhancement on our satellite dish dataset, which we have divided into two subsets: one for training and one for validation. Each subset is populated equally with images of satellite dishes captured under various weather conditions. For the training subset, we augment the satellite dishes by scaling, rotating, and merging them with background images from different weather scenarios to create a diverse set of training images. The same augmentation techniques are applied to the validation subset. In both subsets, the labels are associated with the satellite dishes. By doing so, we aim to test extreme scenarios where the conditions of the backgrounds and the satellite dishes differ significantly. Additionally, by separating satellite dishes in each subset, we prevent overfitting by ensuring that the model does not rely solely on the physical shapes of the satellite dishes for classification.

3.4 Evaluation

In this section, we discuss the experiment setup and evaluation metrics, followed by the evaluation results and results discussion.

3.4.1 Evaluation Metrics

Our model employs YOLACT, which focuses on segmentation and classification, in contrast to YOLO-based models, which primarily focus on object detection and classification. For this study, we focus exclusively on the classification loss component across all models.

To simplify the evaluation and enable a fair comparison between different models, we have redefined the task as an image classification problem instead of an object detection problem. This approach isolates and evaluates the classification performance of each model independently, without the influence of localization aspects such as bounding boxes or masks. By doing so, we eliminate the need to include Intersection over Union (IoU) in our evaluation criteria. Regardless of the IoU threshold, our method selects the class with the highest probability, effectively disregarding IoU altogether.

We redefine our evaluation metrics to align with this classification-focused approach. The metrics are defined as follows: $AP = \frac{TP}{TP+FP}$, $mAP = \frac{1}{C}\sum_{i \in C} AP_i$, $Accuracy = \frac{N_{correct}}{N}$, where TP is true positives, FP is false positives, N is the number of images,$N_{correct}$ is the number of images that get the correct prediction, and C is the number of objects classes.

The YOLO-based models utilize a composite loss function, including classification through cross-entropy, scaled by a factor α, where $\alpha = \frac{\mathrm{C}}{80} \times \frac{3}{\mathrm{nl}}$ which depends on the C and nl.

For our evaluation, we normalize the classification losses from the YOLO-based models by dividing them by α, adjusting them to a comparable scale. Similarly, for our model, we focus on the classification loss $L(y, \hat{y})$, to ensure consistency in the evaluation criteria between all models.

3.4.2 Experimental Setup and Implementation Details

Here we evaluate the proposed method in comparison with the YOLOv7 [29], YOLOv9 [31], R-YOLO [32], and Faster R-CNN models, conducting experiments in Google Colab on an NVIDIA A100 GPU. We did not apply parameter optimization techniques, such as

genetic algorithms, as we want to maintain consistent configurations across all models and avoid extensive parameter tuning to ensure comparability.

We used the Faster R-CNN [25] model from the Detectron2 library and configured it with ResNet50 and Feature Pyramid Networks (FPN). Our experiments utilized a batch size of 8 and ran for 11,000 iterations, which is approximately equivalent to 500 epochs. Learning rate adjustments were applied at 70% and 90% of the total iterations, following a step-wise learning rate schedule. Other settings, including anchor sizes, aspect ratios, and region proposal parameters, remained at their default values.

For comparison with the latest YOLO-based model, YOLOv9, we used the YOLOv9-m model (referred to as "YOLOv9" for short). In addition, we introduced a YOLOv9 variation called YOLOv9-Freeze, in which we applied the partial freezing technique to freeze the backbone of the YOLOv9-m model. The backbone of YOLOv9 is responsible for feature extraction, and by freezing it, we retain its pre-trained capabilities while allowing the head, which handles object classification, to be fine-tuned for the specific task. As partial freezing is used in our proposed model to enhance performance with limited data, YOLOv9-Freeze can be considered directly comparable to the proposed model.

Our study focuses on multi-class classification tasks, evaluating our model on satellite dishes under various weather conditions, and all images used in our analysis are 300 px $\times$300 px in size. We first analyzed scenarios distinguishing between snow-covered and normal conditions, which is essentially a binary classification task. We then extended our analysis to include wet conditions, broadening our model to a more comprehensive multi-class classification.

For the classification tasks, we used model $M_{\text{ResNet50+FC}}$ and fine-tuned it on the weather dataset to distinguish between different weather conditions. The training parameters for $M_{\text{ResNet50+FC}}$ include the Adam optimizer with an initial learning rate of 2×10^{-4}, and weight decay of 5×10^{-4} to optimize the model performance.

To pre-process satellite dish images, we employed YOLACT and used the Adam optimizer with a learning rate of 2×10^{-3} over 30,000 epochs to fine-tune its weights, ensuring the model focuses on the nuanced features of satellite dishes.

3.4.3 Experimental Results

The outcomes of our experiments highlight the classification capabilities of the proposed model compared to YOLOv7, YOLOv9, YOLOv9-Freeze, R-YOLO, and Faster R-CNN. We analyzed the impact of the volume of training images on model performance through the evaluation process.

Based on the data preparation discussed in Section III, we constructed three distinct datasets for each scenario. In the initial scenario, each dataset was paired with a constant test set of 120 images. In the extended scenario, we utilized a larger test set of 180 images to provide a more comprehensive evaluation. Tables 3.1-3.2 summarize the performance of each model in comparison to the baseline methods.

We used four background conditions: snow, sunny, cloudy, and rain. To maintain consistency, each dataset contained an equal number of images for every combination of background conditions and satellite dish conditions. In the initial scenario, this resulted in eight combinations by pairing the four background conditions with two types of satellite dishes. For the dataset with 40 images, we generated five images for each combination. For the dataset with 64 images, we generated eight images per combination. For the dataset with 80 images, we generated 10 images per combination.

In the extended scenario, we added the wet condition satellite dish, leading to 12 combinations. For the dataset containing 60 images, we generated five images for each combination. For the dataset with 120 images, we generated 10 images per combination. For the dataset with 180 images, we generated 15 images per combination.

We compared our proposed model with YOLOv7, YOLOv9, YOLOv9-Freeze, R-YOLO, and Faster R-CNN using the datasets we created. The evaluation metrics for each result were recorded upon completion of the training cycles.

3.4.4 Initial Scenario

Table 3.1 shows that our model outperforms YOLOv7, YOLOv9, YOLOv9-Freeze, R-YOLO, and Faster R-CNN in terms of mAP and accuracy within the initial scenario, which distinguishes between two classes: snow and normal. With 40 training images, our model achieves an accuracy of 78.33%, followed by R-YOLO's 63.33% and YOLOv9-Freeze's 61.66%. With 80 training images, our model reaches an accuracy of 88.33%, compared to 80.00% for Faster R-CNN and 74.16% for R-YOLO.

3.4.5 Extended Scenario

Table 3.2compares the performance of models in multi-class classification, specifically among three classes: snow, wet, and normal. Our model consistently demonstrates superior performance across different sizes of training sets. With 60 training images, our proposed model achieves an accuracy of 66.66%, followed by Faster R-CNN's 59.44% and

Table 3.1: Performance Comparison in the Initial Scenario: Snow and Normal Conditions

Model	Training Images	mAP	Accuracy
YOLOv7	40	24.78%	49.16%
YOLOv9	40	45.55%	46.66%
YOLOv9-Freeze	40	69.25%	61.66%
R-YOLO	40	64.51%	63.33%
Faster R-CNN	40	75.84%	55.83%
Proposed Model	40	**77.69%**	**78.33%**
YOLOv7	64	47.00%	48.33%
YOLOv9	64	52.51%	52.50%
YOLOv9-Freeze	64	69.02%	67.50%
R-YOLO	64	69.14%	68.16%
Faster R-CNN	64	67.77%	63.33%
Proposed Model	64	**82.13%**	**85.00%**
YOLOv7	80	58.77%	51.66%
YOLOv9	80	64.44%	53.33%
YOLOv9-Freeze	80	65.55%	66.66%
R-YOLO	80	74.22%	74.16%
Faster R-CNN	80	82.29%	80.00%
Proposed Model	80	**87.28%**	**88.33%**

R-YOLO's 49.44%. As the training dataset expands to 180 images, our model reaches an accuracy of 88.33%, while Faster R-CNN remain in second place at 81.11%, and R-YOLO comes in third at 72.22%.

3.4.6 Discussion of Model Superiority

Table 3.1 and Table 3.2 show that our proposed model consistently outperforms R-YOLO, Faster R-CNN, and other state-of-the-art models. Our application of the proposed TL method divides the overall loss function in (3.3) into two components: L_{YOLACT} and L^*. The segmentation component, represented by L_{YOLACT}, isolates the target object from its background, allowing for more focused classification. The other component, L^*, represents the loss of a customized core classification for our specific task. Due to the reduced MMD, L^* is already closer to the optimal configuration for our targeted classification task. There-

Table 3.2: Performance Comparison in the Extended Scenario: Snow, Wet, and Normal Conditions

Model	Training Images	mAP	Accuracy
YOLOv7	60	18.90%	33.88%
YOLOv9	60	37.27%	34.44%
YOLOv9-Freeze	60	45.00%	43.88%
R-YOLO	60	53.86%	49.44%
Faster R-CNN	60	58.89%	59.44%
Proposed Model	60	**67.25%**	**66.66%**
YOLOv7	120	33.88%	36.66%
YOLOv9	120	35.12%	37.22%
YOLOv9-Freeze	120	51.88%	52.22%
R-YOLO	120	57.58%	60.55%
Faster R-CNN	120	72.89%	72.77%
Proposed Model	120	**75.26%**	**77.22%**
YOLOv7	180	23.95%	35.55%
YOLOv9	180	37.27%	40.55%
YOLOv9-Freeze	180	61.52%	55.55%
R-YOLO	180	71.57%	72.22%
Faster R-CNN	180	79.75%	81.11%
Proposed Model	180	**86.46%**	**88.33%**

fore, fine-tuning L^* becomes straightforward, focusing on minor adjustments to adapt the model to the target task, which reduces the need for large training data.

In comparison, single-shot models such as YOLOv7 and YOLOv9 show lower performance due to their reliance on large and diverse datasets, as discussed by Zhang *et al.* [37]. YOLOv9-Freeze achieves higher accuracy than YOLOv9, as it benefits from its pre-trained backbone, which utilizes knowledge gained from the COCO dataset to enhance feature extraction capabilities. This use of TL enables YOLOv9-Freeze to require fewer training images, resulting in higher accuracy. R-YOLO benefits from FCNet's feature calibration modules by effectively aligning features across weather conditions, decreasing the need for large and diverse datasets. Faster R-CNN shows strong results due to its two-stage classification process, which trades off processing time for improved performance.

Table 3.3: Performance Comparison with Satellite Antenna Images: Snow, Wet, and Normal Conditions

Model	mAP	Accuracy
YOLOv7	23.95%	35.40%
YOLOv9	37.83%	44.44%
YOLOv9-Freeze	50.99%	46.51%
R-YOLO	72.27%	62.22%
Faster R-CNN	78.33%	73.33%
Proposed Model	**82.18%**	**86.67%**

3.4.7 Generalizability to Real-World Scenarios

We trained the models using 180 augmented satellite antenna images from the extended scenario and tested them on 45 unaugmented satellite antenna images, 15 per weather condition. As shown in Table 3.3, our proposed model achieved the highest accuracy of 86.67%, followed by Faster R-CNN at 78.33% and R-YOLO at 62.22%. The accuracy of our model decreased by about 1.67% compared to the results for the extended scenario, while the accuracy for Faster R-CNN and R-YOLO dropped by approximately 7.78% and 11.94%. This smaller decrease in accuracy demonstrates that our model effectively generalizes to real-world conditions compared to Faster R-CNN and R-YOLO. YOLOv7 and YOLOv7-Freeze maintained low accuracy, conversely, YOLOv9's slight increase in accuracy suggests that it may perform better with this simpler, unaugmented data.

3.4.8 Analysis of Loss Curves

Loss curves show the efficiency of a model's learning by tracking the reduction in error across training epochs. Fig. 3.3 compares the classification losses between epochs for our model, YOLOv7, YOLOv9, YOLOv9-Freeze, R-YOLO, and Faster R-CNN in both the initial and extended scenarios.

In the initial scenario, our model achieves a classification loss of 0.03 within 50 epochs. In contrast, YOLOv7 and R-YOLO reach minimum losses of 0.62 and 0.03 after 500 epochs, respectively. YOLOv9 reaches a loss of 1.20, while YOLOv9-Freeze reaches a loss of 0.666. Faster R-CNN achieves a low loss of 0.0009. In the extended scenario, our model again achieves a loss of 0.03 within 50 epochs. In comparison, YOLOv7 and R-YOLO reach

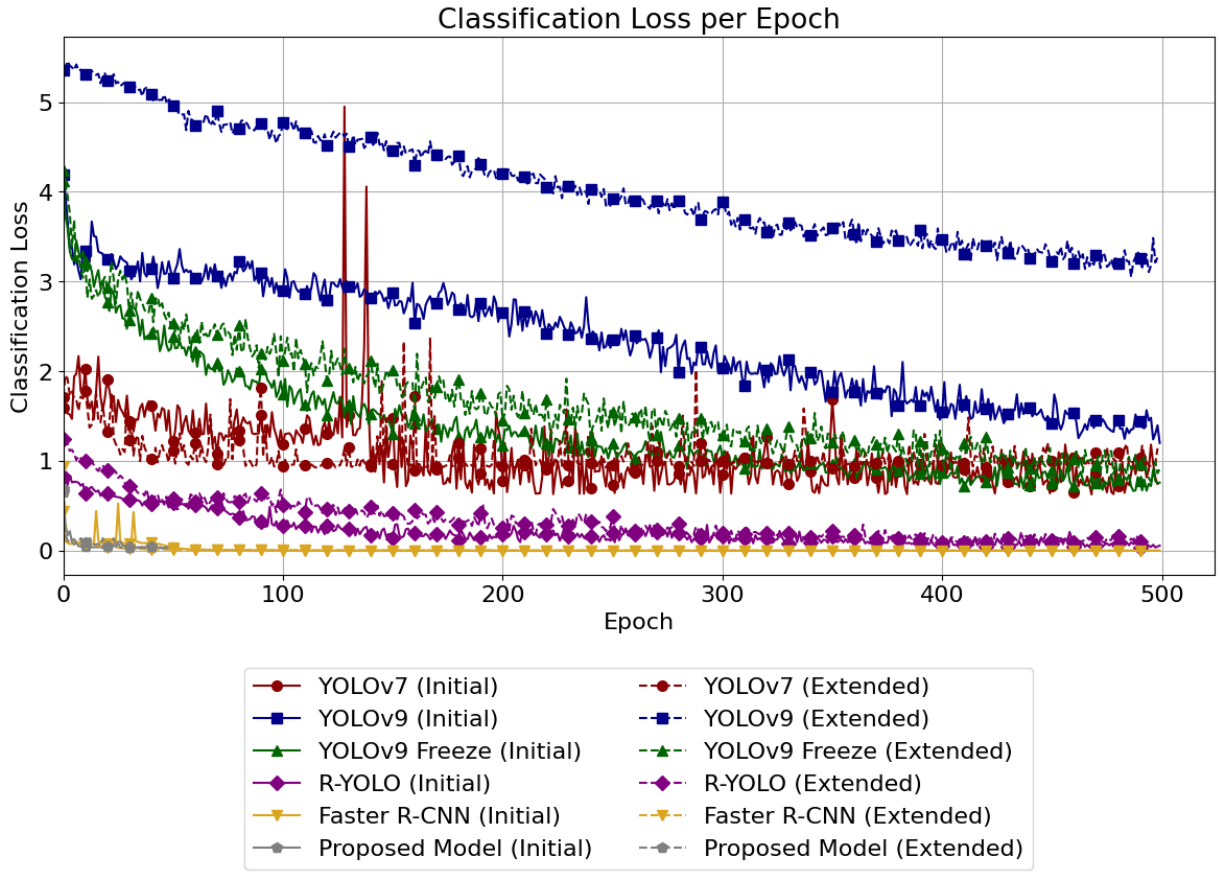


Figure 3.3: Loss curve comparison across epochs for the proposed model and other models. The results are grouped by the "Initial" and "Extended" scenarios.

minimum losses of 0.90 and 0.134 after 500 epochs. YOLOv9 and YOLOv9-Freeze show losses of 3.05 and 0.76, while Faster R-CNN achieves a loss of 0.0011.

In the extended scenario, our proposed model achieves an accuracy of 88.33%, which is higher than the 80.00% accuracy of Faster R-CNN. However, Faster R-CNN achieves a lower classification loss of 0.001, while our model shows a loss of 0.03. The lower classification loss observed in Faster R-CNN suggests a more efficient fitting to the training data. However, it does not fully capture the features required to distinguish satellite dishes. In contrast, our model utilizes TL by incorporating a pre-trained ResNet50, initially trained on simpler weather detection tasks, alongside YOLACT for segmentation. This allows our model to leverage generalized weather detection knowledge and apply it to this specialized classification task. As a result, our model achieves higher accuracy, demonstrating the advantage of TL in adapting to real-world variations.

A similar pattern is observed when comparing our proposed model to R-YOLO in the initial scenario. Our model delivers higher accuracy at 88.33%, compared to R-YOLO's 74.16%, while maintaining a similar classification loss of 0.03.

Table 3.4: Computational Complexity Comparison

Model	**GFLOPs**	**Memory (GB)**
Proposed Model	186.0	16.3
R-YOLO	51.2	14.2
YOLOv7	105.2	4.2
YOLOv9	132.4	28.8
YOLOv9-Freeze	132.4	28.8
Faster R-CNN	238.8	18.5

3.4.9 Computational Complexity Analysis

During the model training phase, we recorded the Giga Floating-Point Operations (GFLOPs) and memory usage. To keep the analysis consistent, we use the input size of 640 px × 640 px for all models, which is the same as the default input size for YOLO-based models and not the default input size of Faster R-CNN model configured on Detectron2 (as Detectron2 uses dynamic input sizes by default). Memory usage was recorded based on a batch size of 16, increasing the batch size would result in higher memory usage.

Table 3.4 shows the computational complexity, particularly focusing on GFLOPs and memory consumption.

The total GFLOPs value for our model is calculated as:

$$\begin{aligned} \text{GFLOPs}_{\text{total}} = \text{GFLOPs}_{\text{YOLACT}} + \text{GFLOPs}_{\text{remover}} + \\ \text{GFLOPs}_{\text{M}_{\text{ResNet50+FC}}} \end{aligned} \tag{3.4}$$

Where $\text{GFLOPs}_{\text{YOLACT}}$ refers to the GFLOPs used by the YOLACT model, $\text{GFLOPs}_{\text{remover}}$ refers to the GFLOPs consumed by the mask remover, and $\text{GFLOPs}_{\text{M}_{\text{ResNet50+FC}}}$ refers to the GFLOPs used by the prediction model.

The GFLOPs value of our model is calculated as the sum of the GFLOPs consumed by YOLACT, the mask remover, and $\text{M}_{\text{ResNet50+FC}}$. For our specific implementation, YOLACT contributes significantly to the total GFLOPs of 118.6, making it the primary factor in the model's computational complexity. The $\text{M}_{\text{ResNet50+FC}}$ component requires 67.4 GFLOPs, while the mask remover adds an insignificant amount of GFLOPs because its operations are straightforward and scales linearly with the image size. In total, this amounts to approximately 186.0 GFLOPs. In comparison, R-YOLO requires 51.2 GFLOPs, YOLOv7 requires 105.2 GFLOPs, YOLOv9 requires 132.4 GFLOPs, and Faster R-CNN requires 238.8 GFLOPs.

In terms of memory consumption, the memory usage during the training phase is determined by the most memory-intensive component in the series of operations. The memory usage for our model is calculated as:

$$\text{Mem}_{\text{total}} = \max(\text{Mem}_{\text{YOLACT}}, \text{Mem}_{\text{remover}}, \text{Mem}_{\text{M}_{\text{ResNet50+FC}}}) \tag{3.5}$$

In (5), $\text{Mem}_{\text{YOLACT}}$ refers to the memory used by the YOLACT model, $\text{Mem}_{\text{remover}}$ denotes the memory consumed by the mask remover, and $\text{Mem}_{\text{M}_{\text{ResNet50+FC}}}$ represents the memory used by the prediction model. Our model requires 16.3 GB of memory, which is higher than R-YOLO's 14.2 GB and YOLOv7's 4.2 GB but lower than YOLOv9's 28.8 GB and Faster R-CNN's 18.5 GB.

While YOLACT provides solid segmentation capabilities, its heavy computational and memory demands present a trade-off. This trade-off is reflected in our model's accuracy and mAP metrics. There is potential for optimizing the model by replacing YOLACT with a simpler image segmentation tool. Such a substitution could reduce both the GFLOPs and memory usage. This possibility can be explored in future work, where the trade-offs between segmentation quality and computational efficiency are carefully balanced.

3.4.10 Discussion of Deployment Options

A general commercial off-the-shelf camera that can generate at least 300 px × 300 px images (which is the image size we used in the section) with its field of view covering the ground terminal being monitored will suffice. To install, the camera should be mounted in a fixed position, ensuring its view consistently captures the entire satellite antenna. Calibration may be needed to adjust angles and avoid interference from environmental factors such as lighting or obstacles. Once mounted and calibrated, the camera can operate continuously or on a scheduled basis. Although we already discussed the scenarios with representative weather conditions, our proposed method with image pre-processing and $M_{\text{ResNet50+FC}}$ model can be extended to work in various forms factors of satellite antennas (e.g., square phased array antennas, large parabolic antennas on Earth stations etc.) and scenarios with additional weather conditions. Due to the generalization of the method, it can also be applied to telescopes used in an optical ground station, which is expected to be used in future satellite Internet.

It is worth noting that in real-world deployments the system may encounter uniquely designed antennas that differ significantly from the samples used to train the YOLACT models. In such cases, the system might struggle to classify the weather conditions on these antennas accurately, potentially reducing detection accuracy. A solution to this would be expanding the training dataset to include a broader variety of antenna designs.

While we utilized an NVIDIA A100 GPU for training and testing, the proposed method can still operate on lower-end GPUs. A less powerful GPU will primarily affect inference time, and if the system's memory capacity is limited, reducing the batch size during inference can help relax memory requirements.

Chapter 4

Final Remarks

This paper addressed critical challenges in SOD and ground terminal performance under adverse conditions. In the first section, we proposed four novel models: GELAN-ViT, GELAN-ViT-Reduced, GELAN-RepViT, and GELAN-RepViT-Reduced. These models effectively combine the strengths of CNNs and ViTs, achieving competitive accuracy on the VOC 2012 dataset while maintaining lower computational costs compared to state-of-the-art deep learning models. Although GELAN-ViT demonstrated superior performance, its high parameter count highlights opportunities for further optimization. Future work will explore strategies to streamline the architecture, enhancing computational efficiency and adaptability for resource-constrained satellite platforms.

In the second section, we introduced a TL-based method to address challenges arising from limited data availability. This approach demonstrated superior performance compared to leading models such as YOLOv7, YOLOv9, R-YOLO, and Faster R-CNN in constrained data scenarios. By effectively handling diverse weather conditions with minimal data, the method showcases its potential for deployment in real-world satellite systems, particularly in environments where large datasets are difficult to collect.

These contributions represent a step forward in addressing the pressing challenges of stability and resilience of the satellite internet. By tackling the issues of efficient SOD and weather condition detection, this work bridges the gaps in satellite communication technology. Looking ahead, these advancements lay the foundation for developing next-generation satellite systems capable of meeting the demands of an increasingly interconnected and data-driven world, while ensuring the long-term sustainability of space operations.

References

[1] Gbeminiyi Ajayi. Multi-class weather dataset for image classification, 2018. Accessed on 2024-03-15.

[2] Q. A. Al-Haija, Manaf Gharaibeh, and A. Odeh. Detection in adverse weather conditions for autonomous vehicles via deep learning. *AI*, 2022.

[3] Martin Aubard, László Antal, Ana Madureira, and Erika Ábrahám. Knowledge distillation in yolox-vit for side-scan sonar object detection. *arXiv:2403.09313 [cs.CV]*, 2024.

[4] D. Bolya, C. Zhou, F. Xiao, and Y. Lee. Yolact: Real-time instance segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9156–9165, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society.

[5] Zhaowei Cai, Quanfu Fan, Rogério Schmidt Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, 2016.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv:2010.11929 [cs.CV]*, 2020.

[7] Johan Garcia, Simon Sundberg, Giuseppe Caso, and Anna Brunstrom. Multi-timescale evaluation of starlink throughput. In *Proceedings of the 1st ACM Workshop on LEO Networking and Communication*, LEO-NET '23, page 31–36, New York, NY, USA, 2023. Association for Computing Machinery.

[8] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv:2107.08430 [cs.CV]*, 2021.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE CVPR*, pages 580–587, 2014.

[10] Dayan Guan, Jiaxing Huang, Aoran Xiao, Shijian Lu, and Yanpeng Cao. Uncertainty-aware unsupervised domain adaptation in object detection. *IEEE Transactions on Multimedia*, 24:2502–2514, 2021.

[11] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE CVPR*, pages 770–778, 2015.

[12] Dominic Laniewski, Eric Lanfer, Bernd Meijerink, Roland van Rijswijk-Deij, and Nils Aschenbruck. Wetlinks: a large-scale longitudinal starlink dataset with contiguous weather data. *arXiv:2402.16448 [cs.NI]*, 2024.

[13] Jiaxu Leng and Y. Liu. An enhanced ssd with feature fusion and visual reasoning for object detection. *Neural Computing and Applications*, pages 1–10, 2019.

[14] Cheng-Hung Lin, Chen-Hsiung Liu, Lungsheng Chien, and Shih-Chieh Chang. Accelerating pattern matching using a novel parallel algorithm on gpus. *IEEE Transactions on Computers*, 62:1906–1916, 2013.

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[16] J.-C Liou, Anilkumar A K, Benjamin Virgili, Toshiya Hanada, Holger Krag, Hugh Lewis, M. Raj, M Rao, A Rossi, and Ram Sharma. Stability of the future leo environment – an iadc comparison study. 04 2013.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[18] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022.

[19] Takeru Oba and N. Ukita. Instance segmentation by semi-supervised learning and image synthesis. *IEICE Trans. Inf. Syst.*, 103-D:1247–1256, 2020.

[20] University of Minnesota. Snow pile classification3 dataset, jun 2023.

[21] Ramakrishna Padmanabhan, Aaron Schulman, Dave Levin, and Neil Spring. Residential links under the weather. In *Proc. of the ACM SIGCOM*, page 145–158, New York, NY, USA, 2019. ACM.

[22] Rui Qian, Robby T. Tan, Wenhan Yang, Jiajun Su, and Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[24] Aref Miri Rekavandi, Shima Rashidi, Farid Boussaid, Stephen Hoefs, Emre Akbas, and Mohammed bennamoun. Transformers in small object detection: A benchmark and survey of state-of-the-art, 2023.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[26] Yunhang Shen, Rongrong Ji, Zhiwei Chen, Xiaopeng Hong, Feng Zheng, Jianzhuang Liu, Mingliang Xu, and Qi Tian. Noise-aware fully webly supervised object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11323–11332, 2020.

[27] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv:1703.00810 [cs.LG]*, 2017.

[28] Aristeidis Tsaris, Chengming Zhang, Xiao Wang, Junqi Yin, Siyan Liu, Moetasim Ashfaq, Ming Fan, Jong Youl Choi, Mohamed Wahib, Dan Lu, Prasanna Balaprakash, and Feiyi Wang. Sequence length scaling in vision transformers for scientific images on frontier, 2024.

[29] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *2023 IEEE/CVF CVPR*, pages 7464–7475, 2023.

[30] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information. *arXiv:2402.13616 [cs.CV]*, 2024.

[31] Chien-Yao Wang, I-Hau Yeh, and Hongpeng Liao. Yolov9: Learning what you want to learn using programmable gradient information. *arXiv:2402.13616v2 [cs.CV]*, 2024.

[32] Lucai Wang, Hongda Qin, Xuanyu Zhou, Xiao Lu, and Fengting Zhang. R-yolo: A robust object detector in adverse weather. *IEEE Transactions on Instrumentation and Measurement*, 72:1–11, 2023.

[33] Yulin Wang, Gao Huang, Shiji Song, Xuran Pan, Yitong Xia, and Cheng Wu. Regularizing deep networks with semantic data augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3733–3748, 2022.

[34] Wen Xu, Jing He, and Yanfeng Shu. Transfer learning and deep domain adaptation. In Marco Antonio Aceves-Fernandez, editor, *Advances and Applications in Deep Learning*, chapter 3. IntechOpen, Rijeka, 2020.

[35] Wenxuan Zhang and Peng Hu. An efficient and generalizable transfer learning method for weather condition detection on ground terminals. *IEEE Transactions on Aerospace and Electronic Systems*, pages 1–8, 2024.

[36] Wenxuan Zhang and Peng Hu. Sensing for space safety and sustainability: A deep learning approach with vision transformers. In *2024 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, FL, US, 16–18 December 2024.

[37] Zhishuai Zhang, Siyuan Qiao, Cihang Xie, Wei Shen, Bo Wang, and Alan Yuille. Single-shot object detection with enriched semantics. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5813–5821, 2017.

[38] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.