Real-Time Implementation and Evaluation of Object Detection and Obstacle Avoidance Systems for UAVs on NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU

Tsedenya Demeke Feissa

A Thesis Submitted to Department of Electronics and
Communication Engineering
College of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's
in Electronics and Communication Engineering
(Electronics Engineering)

Office of Graduate Studies
Adama Science and Technology University

May, 2025

Adama, Ethiopia

Real-Time Implementation and Evaluation of Object Detection and Obstacle Avoidance Systems for UAVs on NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU

Tsedenya Demeke Feissa

Major advisor: Dr. Dereje Tekilu

Co-advisor: Prof. Balachandran Ruthramurthy

A Thesis Submitted to Department of Electronics and Communication Engineering
College of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Electronics and Communication Engineering
(Electronics Engineering)

Office of Graduate Studies

Adama Science and Technology University

May, 2025

Adama, Ethiopia

DECLARATION

I hereby declare that this Master Thesis entitled "***Real-Time Implementation and Evaluation of Object Detection and Obstacle Avoidance Systems for UAVs on NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU***" is my original work. That is, it has not been submitted for the award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Tsedenya Demeke Feissa		
Name of student	Signature	Date

RECOMMENDATION

We, the advisors of this thesis, hereby certify that we have read the final version of the thesis entitled "***Real-Time Implementation and Evaluation of Object Detection and Obstacle Avoidance Systems for UAVs on NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU***". Prepared under our guidance by **Tsedenya Demeke Feissa** submitted in partial fulfillment of the requirements for the degree of Mater's of Science in Electronics Engineering. Therefore, we recommend the submission of the final revised version of the thesis to the department following the applicable procedures.

Dr. Dereje Tekilu	_____	_____
Major Advisor	Signature	Date
Prof. Balachandran Ruthramurthy	_____	_____
Co-advisor	Signature	Date

APPROVAL PAGE OF M.SC. THESIS

We, the advisors of the thesis entitled "***Real-Time Implementation and Evaluation of Object Detection and Obstacle Avoidance Systems for UAVs on NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU*** hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. Dereje Tekilu	_____	_____
Major Advisor	Signature	Date
Prof. Balachandran Ruthramurthy	_____	_____
Co-advisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by **Tsedenya Demeke Feissa** have read and evaluated the thesis entitled "***Real-Time Implementation and Evaluation of Object Detection and Obstacle Avoidance Systems for UAVs on NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU***" and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Electronics Engineering.

_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Final approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and College Graduate Committee (CGC).

_____	_____	_____
Department Head	Signature	Date
_____	_____	_____
College Dean	Signature	Date
_____	_____	_____
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGEMENT

Above everything, I would like to thank Almighty God and the Blessed Virgin Mary for everything I have overcome and passed throughout. This would not have been possible without their blessing, grace and support.

I would like to extend my sincere gratitude and deep appreciation to my advisor, Dr. Dereje Tekilu, for his ongoing unconditional guidance and patience throughout the duration of this study and his insightful contributions as well as material support throughout the research.

I would also like to offer my sincere respect and gratitude to Prof. Balachandran Ruthramurthy for his assistance.

I would again like to extend my appreciation to my family for being next to me from the start to the end of this thesis and for being there whenever I needed something.

At last, I would like to express my genuine thanks to the Adama Science and Technology University (ASTU) for granting me this MSc. Scholarship and finish such an interesting scholarly thesis under the institution.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AI	Artificial Intelligence
AIPC	Aircraft Illustrated Parts Catalog
API	Application Programming Interface
C2PSA	Cross Stage Partial with Spatial Attention
C2f	Cross Stage Partial Focus
C3K2	Cross Stage Partial Bottleneck with Kernel Size 2
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DMA	Direct Memory Access
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
FPS	Frames Per Second
GFLOPS	Giga Floating-Point Operations Per Second
GPS	Global Positioning System
GPU	Graphics Processing Unit
IEMA	Improved Efficient Multi-scale Attention Module
LIDAR	Light Detection and Ranging
LiPo	Lithium Polymer
NPU	Neural Processing Unit
mAP	Mean Average Precision

MASF	Multi-scale Context Aggregation and Scale-adaptive Fusion
MFAM	Multi-scale Feature Aggregation Module
PPO	Proximal Policy Optimization
PVT	Pyramidal Vision Transformers
RL	Reinforcement Learning
SAC	Software Actor-Critic
SoC	System on Chip
SPPF	Spatial Pyramid Pooling - Fast
TPU	Tensor Processing Unit
TRPO	Trust Region Policy Optimization
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
ViTs	Vision Transformers
YBUT	YOLO-based UAV technology
YOLO	You Only Look

ABSTRACT

Current developments in embedded machine learning have unveiled new possibilities in autonomous navigation, especially in the context of unmanned aerial systems (UAVs). With UAVs depending more on onboard systems for real-time perception and reasoning, hardware platform selection is now of utmost importance. This study performs an in-depth benchmarking and performance analysis of embedded edge-computing platforms, i.e., the NVIDIA Jetson Nano, and the Raspberry Pi 4 with Coral TPU, for the deployment of real-time object detection and obstacle avoidance systems in unmanned aerial vehicle (UAV) applications. The central system architecture applies YOLOv11n visual object detection with a ViT-based Proximal Policy Optimization (PPO) reinforcement learning agent to enable semantically driven decision making. All the testing was done in a simulation of the UAV operating in a CoppeliaSim-reconstructed cluttered environment. The work emphasizes key performance metrics such as inference time, energy consumption, reward consistency, and collision rate. These indicators were used to measure real-time performance, computational speed, and the stability of learned policies in obstacle-rich spaces. Experimental results showed that the Jetson Nano, with TensorRT optimization and GPU, consistently reported better performance with lower inference latency of 14.03s and 17% better than the Raspberry Pi with TPU, in terms of collision avoidance, but at the cost of higher energy expenditure of 167J. Conversely, the Raspberry Pi 4 with the Coral TPU provided enhanced energy efficiency but it was outperformed because of its lower inference speed and control action instability with frequent collisions. Overall, this study introduces a novel implementation strategy where YOLOv11n object detection outputs that are contextually enhanced using ViT embeddings prior to input to the PPO in contrast to providing raw image inputs. Additionally, all the models were custom trained and their deployment was validated in CoppeliaSim, integrated via a ZMQ-based remote API. The comparative analysis is the first to benchmark an end-to-end detection and avoidance algorithms for UAV system using real embedded hardware. It provides useful advice to UAV makers, roboticists, and embedded system designers in identification of the edged AI computing devices according to their performance strength.

Keywords: UAV, Object Detection, Obstacle Avoidance, NVIDIA Jetson, Raspberry Pi, Coral TPU, Embedded Systems

CHAPTER ONE
INTRODUCTION

1.1. Background of the study

The revolution of industries such as agriculture, disaster relief, surveillance, and logistics, due to their flexibility and capacity to perform in harsh environments, is attributed to Unmanned Aerial Vehicles (UAVs) or drones (Merei et al., 2025). All these various applications often require that UAVs navigate intricate and dynamic environments independently, where accurate abilities of detecting and evading objects of obstruction are of prime importance in attaining operational success (Rachit Mehul Pathak & Ajay Varma Mudunuri, 2023).

Embedded systems serve as the basic structure of modern UAVs, ensuring they can respond quickly and effectively to changes in surroundings while ensuring safety and mission requirements (KIRAC & ÖZBEK, 2024).The integration of advanced embedded systems within UAVs has made it possible to achieve real-time obstacle avoidance and navigation, providing the computing power needed to process information from multiple sensors and run sophisticated algorithms (Telli et al., 2023; Zhang et al., 2023).

Within the many platforms used in embedded computing within UAS, two of the most notable include NVIDIA Jetson and Raspberry Pi. The NVIDIA Jetson platform is designed for high-compute edge processing, using GPU acceleration that enables it to process computationally intensive functions like deep learning as well as real-time object detection (Choe et al., 2023). Its design is particularly suitable for scenarios that require high throughput, like running deep neural networks like YOLO, which is commonly used in obstacle avoidance (Jetson Nano Developer Kit, 2019). On the other end, the Raspberry Pi platform is renowned for its cost-effectiveness, flexibility, and widespread application in hobbyist and academic use. With its compact design and lowered power consumption, Raspberry Pi has become a preferred choice in scenarios where cost-effectiveness and simplicity outweigh raw computing capability (Mathe et al., 2024).

The Jetson Nano and Raspberry Pi 4 are two of the most commonly matched-up platforms within the increasingly variegated environment of embedded systems for UAV use. The Jetson Nano, based on Maxwell GPU, offers up to 472 GFLOPs of processing capacity, which allows efficient application of deep learning algorithms like YOLO for real-time image processing (Swaminathan et al., 2024). Yet, for its computing power, Jetson Nano is prone to thermal throttling at sustained loads and requires increased power consumption, generally in the range of 5–10W, which is not ideal for battery-powered UAVs. In contrast, the Raspberry Pi 4, not being meant for GPU-intensive operations, provides an inexpensive and power-friendly platform. Equipped with an Edge TPU like Coral USB Accelerator, it is capable of performing quick inferencing on quantized models, finding an optimum power efficiency vs. performance (Baller et al., 2021).

Object detection and avoidance are fundamental capabilities for autonomous UAV navigation, especially in unstructured or natural environments. These are one of the most critical elements of UAV autonomy, which enable object identification and object localization of objects in an image captured by the camera of a drone. Traditional object detection algorithms struggle to work well in aerial image-specific problems, which include variation in height, viewpoint, and small or occluded objects (Wu & Zhou, 2019).

Moreover, YOLO series of algorithms has become prominent for object detection in real time based on its balancing act of speed and accuracy. YOLOv11 brings architectural feature enhancements such as the C3k2 convolutional module and Spatial Pyramid Pooling (SPPF) to enhance feature extraction and detection (C. Wang et al., 2025). Additional modifications, such as Multi-scale Context Aggregation and Scale-adaptive Fusion YOLO (MASF-YOLO) aimed at detecting small objects in UAV imagery, include modules such as the Multi-scale Feature Aggregation Module (MFAM) and Improved Efficient Multi-scale Attention Module (IEMA) (MASF-YOLO: An Improved YOLOv11 Network for Small Object Detection on Drone View, n.d.).

Avoiding obstacles is critical for safe navigation of UAVs in cluttered environments. Conventional approaches tend to be based on predefined rules or reactive agent behaviors,

which do not generalize well to dynamically changing scenarios(Rachit Mehul Pathak & Ajay Varma Mudunuri, 2023) Reinforcement Learning (RL) and, more broadly, Deep Reinforcement Learning (DRL) provides an experience-driven learning method where UAVs learn effective navigation methods by interacting with the environment (Zhang et al., 2023). Proximal Policy Optimization (PPO) is a DRL method that is stable and efficient to use for continuous control problems. Coupled with Vision Transformers (ViTs) that specialize in extracting global contextual information from visual input, PPO is capable of boosting the real-time navigation decision-making capacity of UAVs (C. Wang et al., 2025).

1.2. Problem statement

The utilization of UAV for different sectors has been increasing and the use specific embedded systems is critical. The choice of an efficient onboard system for UAV-based obstacle detection and prevention had been posing many challenges owing to the compromises in computational power, power efficiency, and real-time operations. The NVIDIA Jetson Nano board, while being known for high computational power and support for GPU acceleration features (Jetson Nano Developer Kit, 2019), was often liable for higher power usage, rendering it less suitable for power-limited UAV operations. Although both the platforms are widely used in edge AI-based implementations, there wasn't much comparative analysis, in detail, on their performance in the application for UAV obstacle detection and avoidance. The lack of thorough benchmarking motivated us to fill that research gap through an empirical analysis of the real-time performance and efficiency comparison between NVIDIA Jetson Nano and Raspberry Pi 4 with Coral.

1.3. Objectives

1.3.1. General Objective

To implement and evaluate the performance of edge computing devices for UAV applications with object detection and modified obstacle avoidance algorithms.

1.3.2. Specific Objectives

- To evaluate the performance of custom trained YOLOv11n algorithm implemented on NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU platforms for obstacle detection.
- To evaluate the performance of designed ViTs based RL algorithms implemented on NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU platforms for obstacle avoidance.
- To measure and compare the power consumption and energy efficiency of the two platforms during object detection tasks under simulated UAV flight conditions in CoppeliaSim**.**
- To assess the accuracy and reliability of real-time object detection and avoidance systems on the two platforms hardware validations using simulated UAV environment.

1.4. Significance of study

Through comparison between two projecting edge computing platforms highlighted as being the NVIDIA Jetson Nano and Raspberry Pi 4 with the Coral TPU, this research aims at enhancing UAV system performance. Moreover, it provides practical recommendations for selecting optimized embedded platforms by analyzing significant performance metrics in a simulated UAV environment. The results will help robotics engineers, AI researchers, and UAV developers make informed design decisions for their system creations. The performance evaluation includes machine learning algorithms deployed onto edge devices and tested under dynamic, real-world-like conditions using CoppeliaSim, which has been mainly selected for supporting such condition.

1.5. Scope of study

As UAVs grow more autonomous, the need for efficient onboard object detection and obstacle avoidance becomes non-negotiable. This thesis combines real embedded hardware with a

simulated UAV environment to evaluate performance. Rather than going full physical with drones (and all the chaos that brings), the approach embeds the NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU into a controlled simulation using CoppeliaSim with the implementation of YOLOv11 for object detection and a modified reinforcement learning algorithm for navigation, assessing both platforms on latency, energy efficiency, and decision-making accuracy. The UAV configurations in the simulation are identical which ensures that the comparison stays fair.

1.6. Limitations

This study was affected by numerous constraints, largely caused by platform-specific implementations. The workflows and configurations were closely optimized for once-upfront deployment, which hindered generalizability and ease of reuse. Deployment on the Raspberry Pi 4 with the Coral TPU was limited to those converted to TensorFlow Lite and Edge TPU compilation formats specifically. These formats added constraints in model flexibility and entailed significant preprocessing steps, which were difficult to set in a reproducible manner across environments as well.

Incompatibilities of the necessary libraries and system packages considerably affected the general workflow, usually entailing customized adjustments and versioning reconciliations which hampered the pace of the work. Moreover, the combination of the simulation environment with physical systems using the Remote API also revealed an underlying Sim2Real gap, especially concerning network stability, control feedback latency, and real-time sensory data synchronization. These issues altogether limited the system's extensibility and robustness during a switch from simulation to its possible real-world applications.

1.7. Thesis Organization

This thesis is structured as follows:

Chapter 1 Introduction: This chapter introduces the background and motivation behind

deploying object detection and obstacle avoidance systems on embedded UAV platforms.

Chapter 2 Literature Review: This chapter presents an overview of previous research related to deep learning-based object detection, reinforcement learning for UAV navigation, and the use of edge AI platforms.

Chapter 3 Design and Methodology: This chapter details the architecture of the proposed system, including model selection, simulation setup in CoppeliaSim, training pipeline using ViT + PPO, and implementation on Jetson Nano and Pi 4 + TPU.

Chapter 4 Results and Discussion: This chapter presents the experimental results obtained from real-time deployments, including metrics such as inference time, energy consumption, reward performance, and collision rate.

Chapter 5 Conclusion and Recommendations: The final chapter summarizes the key findings of the research, and provides actionable recommendations for future.

CHAPTER TWO
LITERATURE REVIEW

2.1. Overview

This chapter literature review explores the development and integration of new-age technologies within UAVs, with the focus being on object detection in real-time and autonomous navigation. The chapter starts by following up on YOLO models' development, with a focus on how YOLO adapts to minimal computational resources in a UAV environment. The chapter then addresses reinforcement learning methods, with a focus on how RL can improve autonomous control systems in UAVs. The chapter seeks to explore how ViTs came to change operations in terms of perception in a UAV environment. The chapter also seeks to analyze edge computing hardware for local processing capabilities, breaking dependency on cloud services and latency. The chapter also discusses uses of UAVs, i.e., object detection, avoiding obstacles, and simulation-derived navigation, looking at trends versus challenges. The chapter lastly surveys literature pertaining to a gap in research.

2.2. Progression of YOLO models

2.2.1. Overview of YOLOv1-v11

YOLO series evolved drastically from YOLOv1 to YOLOv11. Every new release came with design improvements with a view to enhancing detection precision, speed, and computational efficiency. The early releases YOLOv1 to YOLOv3 did a fantastic job by introducing object detection in real time with speed-accuracy trade-offs. The subsequent releases YOLOv4 and YOLOv5 brought more advanced methods like part stage-to-stage connection and mosaicking data augmentation to further advance performance. YOLOv6 up to YOLOv11, focused more on model optimization towards deploying to resource-constrained applications without compromising accuracy(Sapkota et al., 2025). However, YOLOv11brought new blocks like C3k2 block and C2PSA module, whose presence enabled enhancing mean Average Precision (mAP) scores by reducing parameter quantities within the model(Khanam & Hussain, 2024).

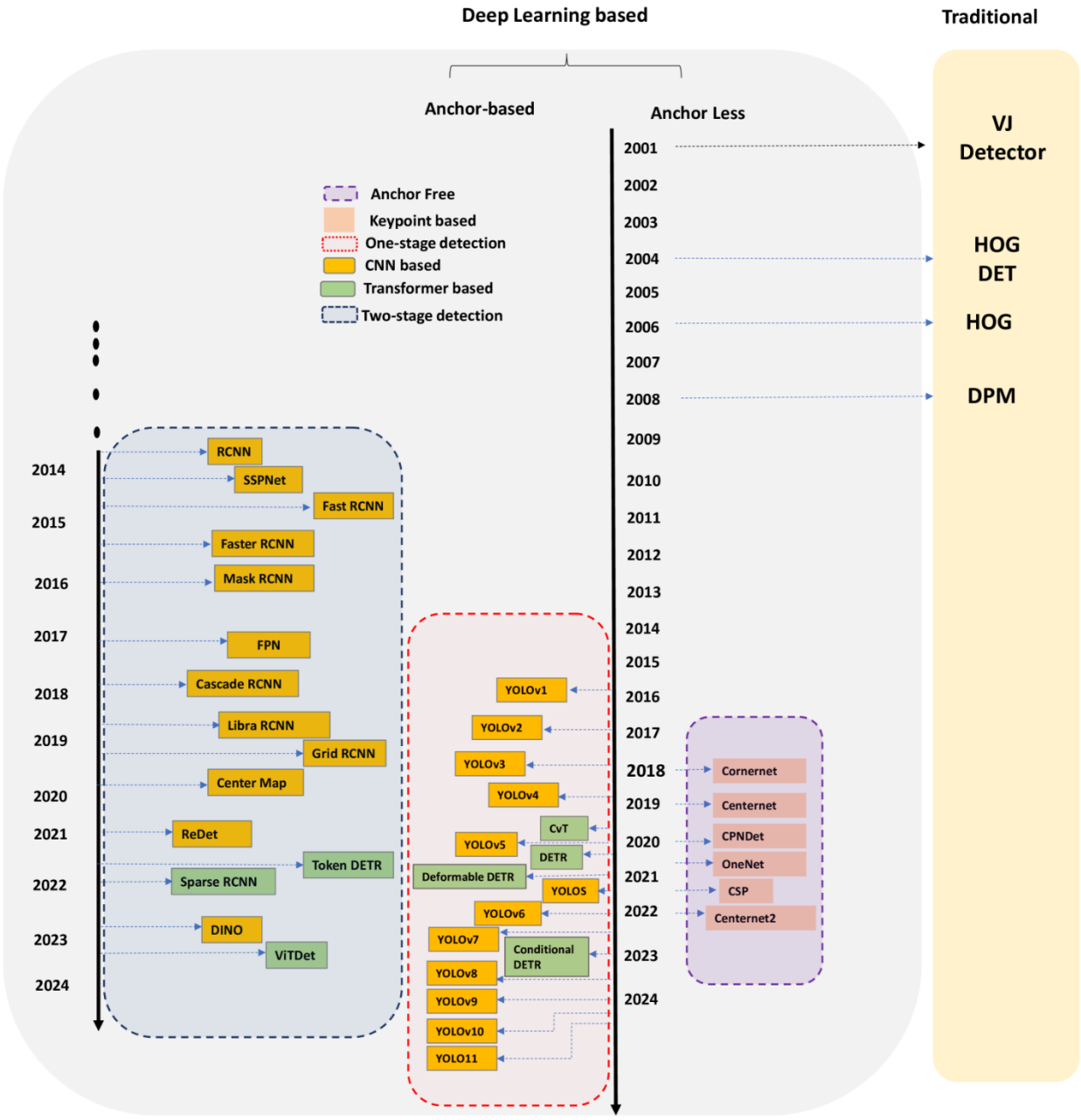

Figure 2.1. Timeline of YOLO advancement (Sapkota et al., 2025)

2.2.2. YOLOv11

YOLOv11 is a major leap in YOLO object detection models with architectural innovations tailored to better detection accuracy, speed, and efficiency. The major innovations include C3K2 blocks replacing C2f modules for enhanced detection efficiency and speed (*YOLO Explained: From v1 to Present*, n.d.). The blocks utilize smaller convolution kernel sizes with an optimized bottleneck structure, which increases efficiency without sacrificing detection accuracy. The C2PSA module also incorporates channel attention and spatial attention mechanisms for enhanced contextual awareness, enabling YOLOv11 to be more deployable across multiple deployment contexts from edge hardware to compute-intensive ones(Khanam & Hussain, 2024; *YOLO Explained: From v1 to Present*, n.d.).

YOLOv11 stands out as the most advanced and efficient version in the YOLO series. Compared to earlier models, it delivers faster inference, improved accuracy, and uses fewer parameters. Designed with edge devices and GPU platforms in mind, YOLOv11 is well-suited for real-time applications such as drone-based object detection(Sapkota et al., 2025).

2.3. Reinforcement Learning for Control

2.3.1. RL Algorithms

Reinforcement learning has come a long way, especially with algorithms like PPO and different Actor-Critic setups. PPO, which was introduced by (Schulman et al., 2017)a policy gradient method that hits a nice balance between being easy to implement and actually performing well. It uses a clipped surrogate objective to keep policy updates in check, which helps make training more stable. That’s one of the big reasons it's become so popular it just works across a bunch of tasks, whether it's controlling robots or playing games(Schulman et al., 2017).

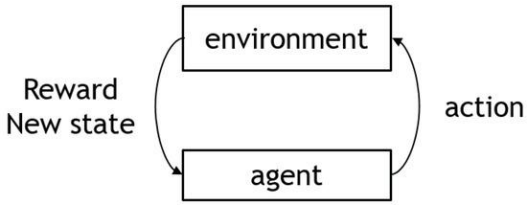

Figure 2.2. Reinforcement Learning Block (D. Han et al., 2023)

The Actor-Critic framework is a smart blend of policy-based and value-based reinforcement learning. Essentially, the actor decides what action to take, whereas the critic evaluates that action by assessing how good it was, then uses that feedback to improve the policy. This setup really improves learning stability and efficiency, especially in environments with continuous action spaces(Romero et al., 2023). Advanced versions, like Soft Actor-Critic (SAC), take it further by using entropy growth, which encourages more exploration and leads to stronger, more robust performance which in turn makes them so effective for handling complex, real-world control problems(Haarnoja et al., 2018).

And next to the zoo of RL algorithms, TD3 playing with its twin critics and SAC chasing entropy and TRPO flexing its math muscles, PPO is a sweet spot between power and usability. It steers clear of TD3's two critic drama and SAC's entropy obsession in favor of clipped objective to keep updates sensible and training stable. TRPO might get its feet wet in second-order optimization with all the grace of a proof on a college board, but PPO does nearly as good a job minus all the computation. It's not the most flashy or theoretically beautiful method, but if you want something that works pretty well across a wide range of continuous control tasks, PPO is a safe bet (Ghasemi et al., 2024).

Deep Q-Networks (DQNs) marked a major shift in reinforcement learning by blending traditional Q-learning with deep neural networks, allowing agents to handle complex, high-dimensional environments. Originally designed for discrete action spaces, DQNs used convolutional neural networks to map raw inputs, like images, directly to Q-values, which proved hugely effective in mastering tasks such as Atari games(Neil de la Fuente & Daniel A. Vidal Guerra, 2024). In 2025, they took the central concept and pushed it into completely new grounds, beginning with self-driving. By integrating the DQNs along with semantic segmentation, the researchers endowed

the car with deeper perception which has, a more intelligent agent to make more intelligent driving choices, with their enhanced Double DQN improving control(Gao et al., 2025; Jabr et al., 2025).

Another algorithm is DDPG, an actor-critic algorithm for a continuous action space, and it finds application in robotics and control. A 2025 paper utilized a variant of SD3 to control a grid-connected inverter to minimize power oscillation in a smart grid with adaptive policy learning(Yang & Wang, 2025). A different study used DDPG to address instability in off-policy evaluation by incorporating a safety-constrained behavior policy to minimize variance and maintain real-world safety. Although successful, DDPG is still prone to the specifics of the exploration strategy and hyperparameter tuning(Chen et al., 2024).

2.3.2. Learning Stability of PPO

The stability of the learning of reinforcement learning algorithms plays an important role in the performance of these algorithms, especially in the presence of a challenging environment. PPO has been praised for achieving both performance and stability during training(Meng et al., 2023). Through the usage of a clipped surrogate objective function, PPO inhibits the policy update step to be too large, preventing large updates leading to performance collapse. The method has been shown to be robust to numerous tasks ranging from robotic control to game playing(Y. Wang et al., n.d.).

Nevertheless, PPO has its shortcomings. It has been shown in various works that although PPO seeks to constrain policy updates in a trust region, it makes no strict attempt to do so, and this can result in instability in some situations(Y. Wang et al., 2019). To overcome this, some improvements were proposed, which update the clipping range adaptively to enhance the exploration and stability. In the time-honored tug-of-war between exploration and exploitation, the proposed improvement of PPO added a double-objective framework balancing the quest for large rewards with the retention of a healthy amount of exploration. By adding clipped surrogate objectives, the approach stabilizes policy updates and prevents sudden oscillations that might revert previously learned hard-won knowledge. What's innovative here is the recognition of

exploration diversity as an integral component of the learning process, getting agents to explore purposefully, not simply wander at large. It prevents convergence to bad strategies too early, a particularly valuable capability in complicated, uncertain situations where flexibility matters(Khoi et al., 2021). It's the equivalent of encouraging an agent playing a volatile game to not only succeed, but to find a mechanism to continue to improve throughout.

2.4. Vision Transformers

Transformers is a powerful library from Hugging Face that gives easy access to state-of-the-art pretrained models for vision, text, audio, and multimodal tasks. It’s super useful whether you're training on your own data or just want fast, plug-and-play inference. It comes with a simple pipeline for common tasks, plus a built-in trainer with features like mixed precision and Flash Attention. It also handles text and vision-language generation really efficiently, which makes it ideal for building smart, responsive applications quickly (*Transformers*, n.d.).

Transformers have rapidly revolutionized computer vision by providing a new paradigm to interpret visual information one that transcends the local limitations of conventional convolutional neural networks. Originally developed for the task of natural language processing, their effective self-attentive mechanism enables global and adaptive feature learning, fitting them to a variety of tasks such as classifying images, objects, and segmenting them(Khan et al., 2022). This paper states the increasing involvement of Transformers in vision and the simplicity, flexibility, and scalability of Transformers to different types of data, including images, text, and video.

Originally developed for natural language processing, Vision Transformers (ViTs) have made a big impact in computer vision, showing strong results in tasks like image classification, segmentation, and detection—and in many cases, even outperforming traditional CNNs thanks to their attention-based design(Fu, 2022). Newer versions like Pyramidal Vision Transformers (PVT), Swin Transformers, and SegFormer have taken things further by adding features like localized attention and hierarchical processing, making these models more efficient and scalable for real-world applications. These improvements enable ViTs to become efficient at dense prediction tasks and handling of high-resolution inputs, both of which are requirements for UAV-

dependent vision systems. ViTs, in the context of reinforcement learning (RL), suffer from being hungry for data and experiencing representation bottlenecks (Tao et al., 2022).

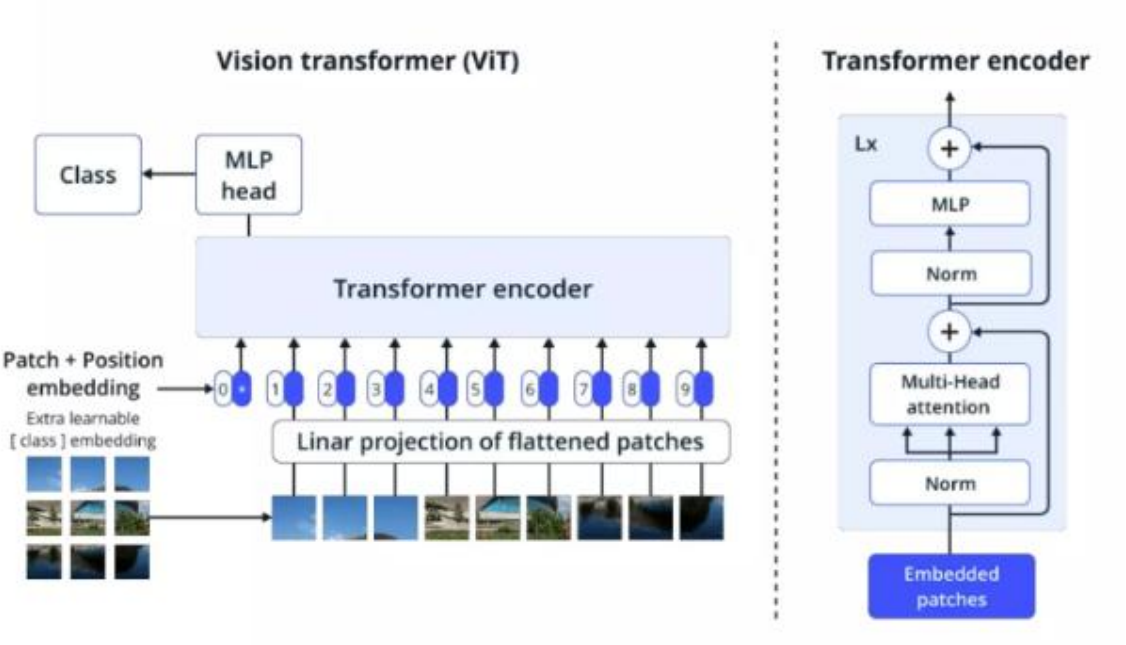

Figure 2.3. Vision Transformer(Benjaminlapid, 2023)

2.5. Edge Computing Devices for UAV Applications

2.5.1. Raspberry Pi 4 for UAV

The Raspberry Pi 4 has turned out to be a surprisingly capable and affordable option for running edge AI on drones, especially when it comes to vision and decision-making tasks. For example, (Ortega et al., 2023)used it alongside a LIDAR and camera module to handle image processing right on the drone, showing that even a small board like the Pi can manage embedded vision (Toma et al., 2022)pushed this concept a step further by actually running the machine-learning models on the Raspberry Pi directly, allowing drones to process visual information in real time without a cloud connection. No longer would decisions lag, latency would be reduced, and the drone would be far more responsive while in flight.

It was taken a closer look by (Albanese et al., 2022) at using the Pi for vision-based navigation, focusing on energy efficiency which is a must for compact drones with limited power. Their work

pointed out how well the Pi balances computing power and energy use, which is crucial for keeping drones in the air longer and making them more reliable. On the security side, (Alam et al., 2019)showed how the Pi could even be used for real-time video analysis to detect abnormal events, proving its value for surveillance and monitoring tasks. Altogether, these studies make a strong case for the Raspberry Pi 4 as a solid fit for UAVs doing edge analytics, so long as the workload stays within its limits, given the board's thermal and power constraints.

2.5.2. NVIDIA Jetson Nano for UAV

The NVIDIA Jetson Nano has quickly become a go-to choice for powering AI-driven UAVs, especially when it comes to real-time inference and computer vision. In one study (Hakani & Rawat, 2024) used it alongside YOLOv9 to detect drones in real time—proving the Nano can handle deep learning models at the edge without breaking a sweat. Likewise, (Ma et al., 2023) utilized it to enrich visual landing capability with YOLO-powered detection to enable drones to land accurately and smoothly, all with no reliance on cloud computing. (Hsieh et al., 2023) further added by demonstrating a lean-weight visual navigation system to support the Nano while maintaining real-time performance in the indoor space, a blessing for small drones with low computing capability.

However, the Jetson Nano isn't just about sight. (Vasishta et al., 2024) showed how it works with small object detection applications to demonstrate efficiency despite limited resources. It pushed further with GPS-denied operations, though, when (Theses et al., 2021) used the Nano to allow UAVs to detect landmarks and map cities autonomously in real time. It's not competing with the best-of-class GPUs, but the Jetson Nano strikes a sweet spot: low-energy, small, and impressively powerful, a perfect match for intelligent, autonomous UAV applications needing edge computing sans the bulk.

2.5.3. Coral TPU for UAV

The Google Coral Edge TPU has been a top choice in UAV development owing to the ability to implement TensorFlow Lite models at lightning speeds while conserving low levels of power.

(Vermesan et al., n.d.) put it to test using YOLOv5 for object recognition in drones and concluded it to be effective even in lighter, smaller platforms. Moreover, (Garcia-Perez et al., 2023) pointed out how the Coral TPU stands out for its fast inference and low energy use, two things that are super important for keeping drones in the air longer. Taking it a step further, (Bemposta Rosende et al., 2023) used the Coral in a real-time UAV traffic control system, and even under tight power constraints, it delivered solid performance, showing how well it handles demanding onboard tasks.

It was also explored how Coral TPUs can enable shared learning across multiple drones. The combination of each UAV with a Coral TPU and allowing them to collaborate through on-device inference led to a decentralized approach to decision-making. This setup not only eases the load on communication networks but also makes drone swarms more scalable and resilient. Taken together, these findings highlight why the Coral TPU is such a strong candidate for real-time, AI-powered UAV missions, where energy efficiency, autonomy at the edge, and low-latency performance really matter (Dubois, n.d.).

2.6. UAV Applications and Navigation

2.6.1. UAV-based Object Detection

Object detection using UAVs has its own challenges. Aerial images typically involve tilted camera views and a variety of altitudes, along with very small objects to detect. To address this, researchers have begun to modify the detection models to accommodate these conditions, emphasizing multi-scale feature representation and spatial attention tailored to the views seen from a drone. Models such as PETNet and EL-YOLO are perfect representatives, integrating prior knowledge with efficient designs to enhance performance in low-altitude situations. These models are designed to accommodate the low resolution and object congestion typically presenting UAV video as challenging to work with, in applications such as environmental surveillance or tactical reconnaissance(T. Wang et al., 2023; Xue et al., 2024).

Moreover, thanks to their ability to capture high-resolution imagery over time and space, drones are ideal for detecting objects in real time, even in complex environments. The use of YOLOv3,

v4, and v5 for detecting people and vehicles in thermal infrared drone footage was explored in (Jiang et al., 2022) . Among them, YOLOv5-s stood out with the highest mAP of 88.69% and an impressive 50 FPS detection speed, making it a strong fit for onboard systems that need to work day and night using thermal sensors. In another study, (Liu et al., 2020) introduced UAV-YOLO, an improved version of YOLOv3 designed specifically for spotting small objects from the air.

It is also interesting how the YOLO family of models has moved well beyond traditional surveillance roles. It is also highlighted how YOLO-based UAV technology (YBUT) is now being used in everything from agriculture and disaster response to automated transport. More recent variations of YOLO, including YOLOv4-Drone, YOLO-Neck, and ViT-YOLO, were tried, which all have been optimized for drone applications. These variations reflect the versatility of the YOLO framework, particularly for secondary development for drone platforms. Tests on specific databases showed YOLO variations to be faster and accurate compared to the rest, confirming their importance for real-time processing on embedded UAV platforms (Chen et al., 2023) All these works describe why YOLO stands first for object detection on drones.

2.6.2. UAV-based Obstacle avoidance

RL is turning out to be a useful mechanism in UAVs obstacle avoidance with more resilience and autonomy. (Skarka & Ashfaq, 2024) compare how the combination of RL with traditional machine learning makes UAV systems more reactive, systems able of responding in real time and planning better. Another compelling perspective comes in the form a paper written by (Yanes Luis et al., 2021)specifically focused on surface vehicles, the point is the same: deep RL approaches provide enhanced generalization and scalability over complicated situations compared to evolutionary algorithms, they apply equally to UAVs as well, particularly in environmental monitoring where the same sensors and navigation issues occur.

In both simulation and real-world tests, deep RL continues to hold potential for UAVs. Singla & (Singla et al., 2021) for instance. They proposed a memory-augmented deep RL model allowing drones to navigate using only monocular vision, when they do not even possess complete environmental information. It’s perfect for cluttered, unstructured airspace.

2.7. Related Works

The performance of YOLOv5 and YOLOv8 for UAV object detection when used on a Raspberry Pi 4 is examined in a paper by (Karsan Marco Giordani et al., n.d.). Both models were evaluated using optimization frameworks like OpenVINO and NCNN, with a focus on making them competent for edge devices, using widely used standard datasets like Stanford Drone and VisDrone. The experiment focused on close examination of the major factors of inference time, energy consumption, and detection precision (mAP). The performance test proved YOLOv8's lighter models, to be exceptionally accurate at 56.1% mAP while maintaining low latency, making it a top contender for drone processing.

A study by (Rachit Mehul Pathak & Ajay Varma Mudunuri, 2023) compares two computer vision approaches, YOLOv5 and traditional Convolutional Neural Networks (CNNs), to see which performs better for autonomous obstacle detection and avoidance in drones. The authors walk through a full end-to-end setup using AirSim and Unreal Engine to simulate and test how well each model handles object detection in aerial environments. They trained the models on annotated images of various obstacles and evaluated them using common metrics like precision, recall, F1-score, and mean average precision (mAP). The results clearly show that YOLOv5 outperformed CNNs in both speed and accuracy, making it a strong choice for real-time drone navigation. The study also brought in LiDAR for 3D mapping and used CNNs as a benchmark to give more context to YOLOv5's performance. Looking ahead, the authors suggest that integrating Deep Reinforcement Learning (DRL) could take things further by enabling more adaptive, intelligent decision-making in complex environments.

On the other hand, (Jayanth et al., 2024) carried out a detailed comparison of edge computing platforms, specifically looking at how CPUs, GPUs, and NPUs handle tasks like linear algebra and neural network inference. Their results showed that NPUs stood out for their energy efficiency and smart data handling (thanks to DMA), which helped them achieve lower latency in tasks like matrix-vector multiplication. In certain inference workloads, even the NPUs outperformed the GPUs. Nevertheless, the GPUs still dominated the workload in more computationally intensive operations such as matrix multiplication in its entirety. The study gives

insights into how heterogeneous system-on-chip (SoC) devices, similar to Intel's AIPC, can find a useful balance between responsiveness, power consumption, and speed. These observations are particularly valuable to consider while making decisions as to where and how to execute AI models in drone applications where every bit of performance and power matters.

Another research is about several popular object detection models, which are, YOLOv8, EfficientDet Lite, and SSD. These are tested on both the Raspberry Pi and Jetson Orin Nano, tested with and without Coral TPUs. They looked at key factors like energy use, inference time, and mAP on the COCO dataset. YOLOv8 Medium provided the best overall accuracy, albeit at the cost of greater power consumption. SSD MobileNet V1, however, was the fastest and the least energy-hungry, thus the best option when every bit of energy counts. Of the available hardware, the Jetson Orin Nano led the pack with the optimal compromise between speed and efficiency, therefore the best option for real-time, energy-sensitive drone applications(Baller et al., 2021).

In this study, the discussion is about the performance of Jetson Orin NX, Jetson Nano, and Raspberry Pi 5 with Coral TPU across AI workloads using YOLOv5. It mainly focus on practical deployment factors like frame rate, power consumption, and thermal behavior. The Jetson Orin NX delivered almost double the inference speed compared to the Pi 5 with Coral TPU, but it came with higher power demands. For UAV applications, where both efficiency and compact size are key, the findings are super useful. They give a clear picture of the trade-offs between raw performance and energy usage, which is exactly the kind of insight that matters when working with lightweight drones that need to manage power and heat while still handling real-time vision tasks(Minott et al., 2025).

A paper by (Baller et al., 2021) presents an effective framework for benchmarking edge devices such as the Jetson Nano and Coral Dev Board based on factors like inference time, power consumption, and support for deep learning models. Their results reflect how the Jetson Nano favors the use of lightweight models through its GPU acceleration while the Coral Dev Board leads with energy efficiency when using quantized TensorFlow models. As far as the UAV setup is concerned, it directly validates the necessity of hardware matching the detection model on the

basis of whether raw performance or saving power takes precedence.

In the work of a (Calderon-Cordova & Sarango, 2023) forcement learning framework was established utilizing the A2C algorithm to train a Franka Emika Panda robot arm for a reach-to-target task within CoppeliaSim. A client–server setup was utilized, where CoppeliaSim was responsible for the handling of physics as well as the display, while the training was performed on the Python side utilizing Gymnasium and Stable Baselines3. The agent was faced with an 18-dimensional observation space, with joint positions, end-effector coordinates, and target distance. The action was mapped directly to the seven angles of the robot. The neural network employed a split structure, with the Actor and Critic subnetworks as different components, each with three layers using ReLU activations

2.7.1. Research Gap

Most of the studies out there tend to look at object detection models or edge devices separately, usually testing them under fairly controlled or general-purpose conditions. What's often missing, though, is attention to the real-world challenges of deploying these systems on UAVs, things like real-time decision-making, handling aerial movement, or adapting to dynamic environments. Very few use simulation platforms like CoppeliaSim to replicate actual drone scenarios, and even fewer test how object detection and decision-making work together on embedded hardware. This study steps in to fill that gap by bringing together object detection, reinforcement learning, and real-time simulation into one integrated setup, offering a more grounded and practical approach for real UAV applications.

Title and Author	Contributions	Identified Gaps
Performance Analysis and Evaluation of Object Detection Algorithms for Drone Networks(Karsan Marco Giordani et al., n.d.)	It compares YOLOv5 and YOLOv8 on Raspberry Pi 4 to perform UAV object detection on setups like Stanford Drone and VisDrone. It employs OpenVINO and NCNN to achieve optimization for deployment on the edge and tests inference time, power consumption, and mAP.	It doesn't have UAV-specific factors such as flight mobility or real-time navigation. It does not incorporate simulation and decision-making. The thesis, alternatively, addresses CoppeliaSim-based navigation and integrates detection and PPO-agents.
Autonomous Obstacle Detection and Avoidance in Drones(Rachit Mehul Pathak & Ajay Varma Mudunuri, 2023)	Pathak and Mudunuri (2023) compare YOLOv5 and CNNs for drone-based obstacle detection using AirSim and Unreal Engine. The study demonstrates that why YOLOv5 outperforms traditional CNNs.	The research does not benchmark real-time deployment on the edge devices and does not evaluate decision-making RL policies and ViT embeddings.
DeepEdgeBench: Benchmarking Deep Neural Networks on Edge Devices(Baller et al., 2021)	Baller et al. (2021) compared object detection models - YOLOv8, EfficientDet Lite, and SSD - on Raspberry Pi and Jetson Orin Nano devices, with and without a Coral TPU. Among devices, Jetson Orin Nano provided the best balance of speed and power efficiency and is a suitable candidate to be used in power-constrained applications like drones.	This does not investigate deployment in environments that are UAV-specific nor does it cover navigation and control complexities. It does not perform simulation testing or real-time UAV application scenarios like this thesis.
DTPPO: Dual-Transformer Encoder-Based Proximal Policy Optimization for Multi-UAV Navigation in Unseen Complex Environments(Wei et al., 2024)	It contributes a Dual-Transformer architecture for PPO that improves multi-UAV coordination and generalization in unseen simulated gym-pybullet-drones environments with random obstacles.	It doesn't incorporate YOLO based object detections. Also, the simulation environment lacks richer real physics, etc.

Table 2.1. Contribution and gaps

CHAPTER THREE
DESIGN AND METHODOLOGY

This study takes a hands-on, real-time approach to building and testing an autonomous UAV system that can handle both object detection and obstacle avoidance. It combines simulation-based training with actual deployment on edge AI hardware. The goal is to see how well-customized deep learning models that are, a hyper-tuned YOLOv11 for object detection, and a modified RL for decision-making can hold up when put to the test when running under real constraints like limited computing power edge computing devices that need low-latency responses, and constantly changing obstacle conditions, just like you'd expect in real UAV missions.

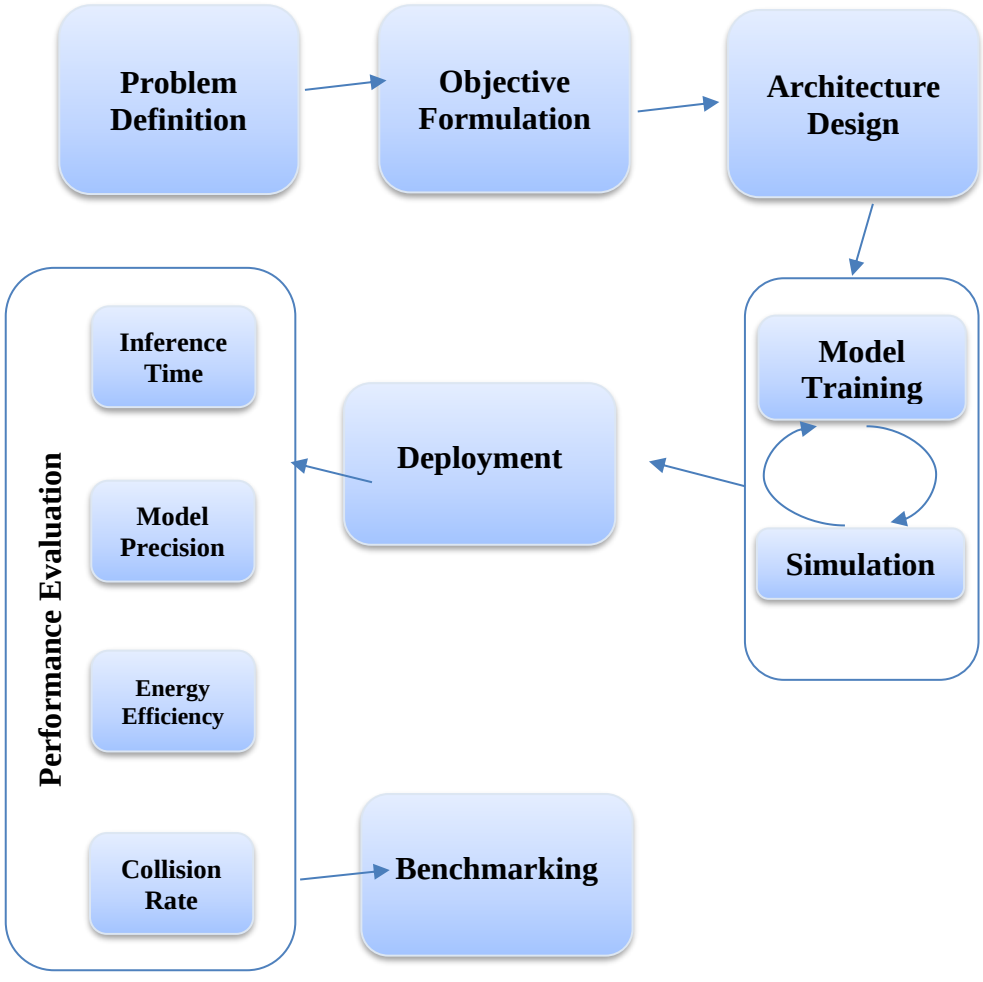

Figure 3.1. Methodology Flow Chart

3.1. Research Method

Here are the main techniques that are based on the overall framework presented in the methodology;

- Specific UAV environment was created in CoppeliaSim with different settings of obstacles to implement replicable, demanding conditions for training and testing.
- Use of both the raw data and the simulation-generated synthetic data was utilized to train the object detection and obstacle avoidance models respectively.
- The trained models subsequently were deployed onto edge AI hardware, the Jetson Nano and Raspberry Pi 4 with the Coral TPU, for the purpose of testing real-time performance.
- The performance of the system was evaluated with measures applicable to actual requirements.

3.2. Tools and Frameworks

The research is based on an integration of simulation tools, AI libraries, and hardware-specific frameworks for constructing and validating the UAV system:

Simulation

The simulation was developed using CoppeliaSim (v4.9) to simulate realistic UAV surroundings with dynamic obstacles and sensor integration.

Programming

The main programming language used for simulation, model training, and deployment management was Python.

Deployment libraries

TensorRT was used to accelerate deep learining inference on Jetson Nano and compiled Edge TFLite was used for the Raspberry Pi with Coral TPU implementation.

3.3. Data Acquisition

3.3.1. Raw Data

To develop a reliable object detection model for UAV navigation, the training data was gathered and curated using Roboflow, a platform that streamlined the processes of dataset management, annotation, and preprocessing. Three separate datasets, each corresponding to one of the target classes, birds, trees, and buildings, were merged to form a unified dataset relevant to obstacle avoidance during flight. In spite of having pre-existing annotations in the datasets, a manual check was performed to fix any incorrect labels or bounding boxes and provide consistency to the dataset. Other images have also been included, specifically for enhancing the coverage of birds in different light conditions and flight angles.

Datasets	*Train*	*Validation*	*Test*	*Total dataset*
Bird	2211	1106	632	3949
Tree	792	306	37	1135
Building	686	195	97	978

Table 3.1. Training dataset

With the refinement of annotation, the augmentation functionality of Roboflow was utilized to simulate realistic conditions typically faced in UAV usage such as, random horizontal flipping, rotation transformation, and changing the perspective of the camera. The dataset was resized, normalized, and YOLO-exported to be compatible with the input requirements of the YOLOv11 model employed in the research.

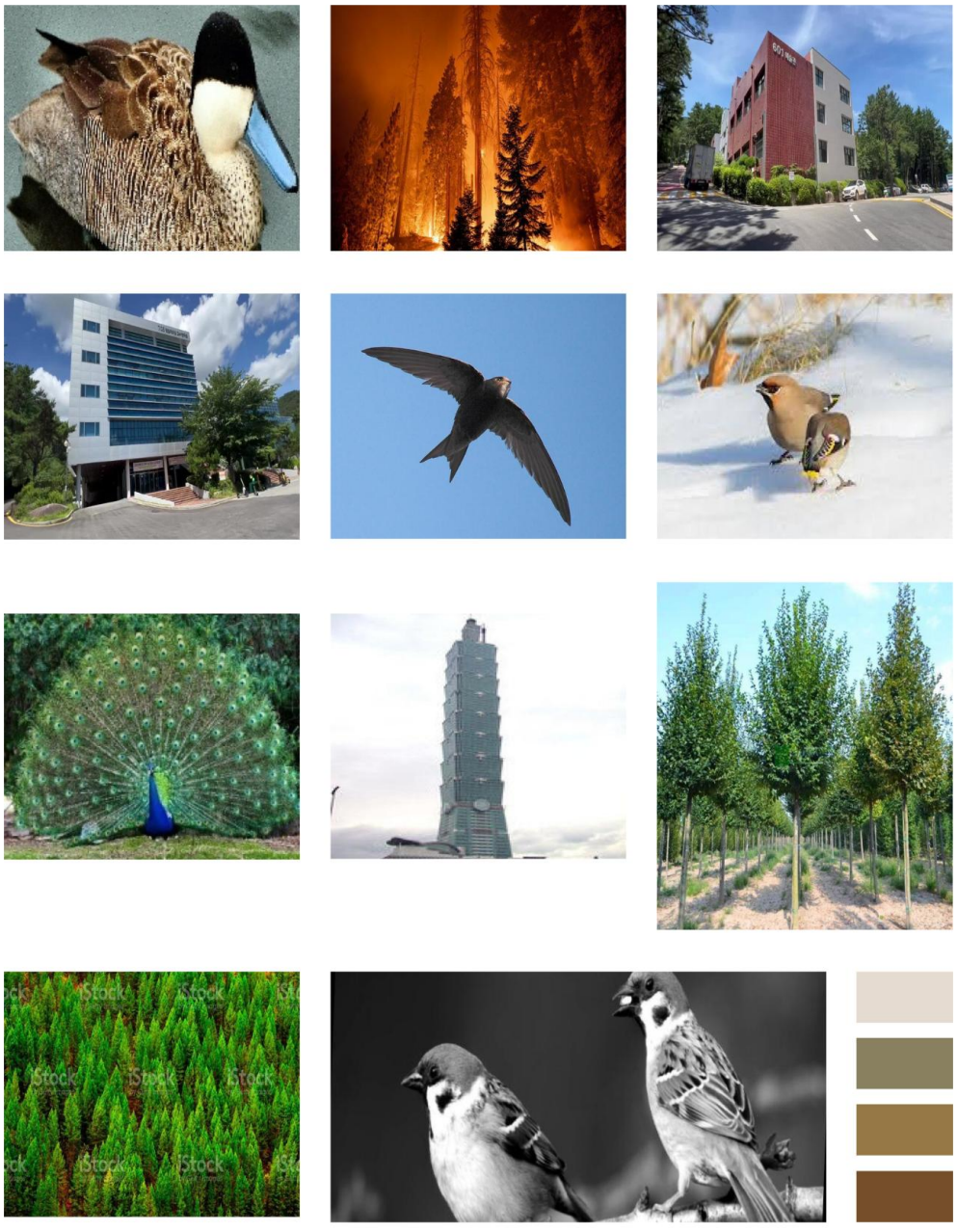

Figure 3.2. Raw datasets

3.3.2. Vector Data

In this UAV applications' real-time obstacle avoidance, the PPO agent does not make decisions from raw images or sensor inputs. It uses vector inputs, organized numerical representations of the world that are more comprehensible and clearer. The data comes from the feature extraction of Vision Transformers. They process the raw camera input and represent it as compact, high-dimensional vectors. The vectors encode critical spatial and contextual information about the world so the agent can comprehend what is occurring in the world surrounding it without becoming overwhelmed with pixel-level information(Fu, 2022).

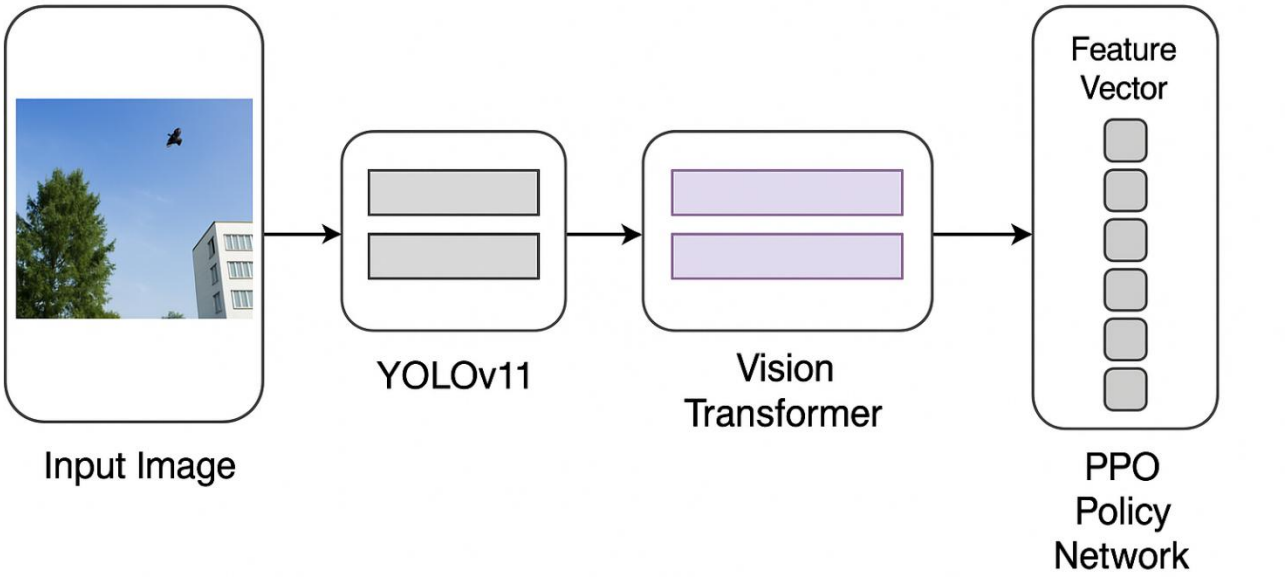

Figure 3.3. Data vectorization process

3.4. System Architecture design

The architecture of the system designed in this research adopts a hybrid model, with a simulation-based training and real-time execution on edge devices for facilitating autonomous UAV navigation functions. With the simulation platform of CoppeliaSim, UAV is positioned in a simulation space with obstacles, birds, trees, and buildings, replicating the conditions under which it would be operating under actual missions. Visual inputs of the UAV are first perceived by the YOLOv11n model and the detected images with bounding box and confidence information

are processed by a ViT which converts the outputs into compact feature vectors. These vectors act as the “state” information for a PPO agent, deciding what action the UAV should take.

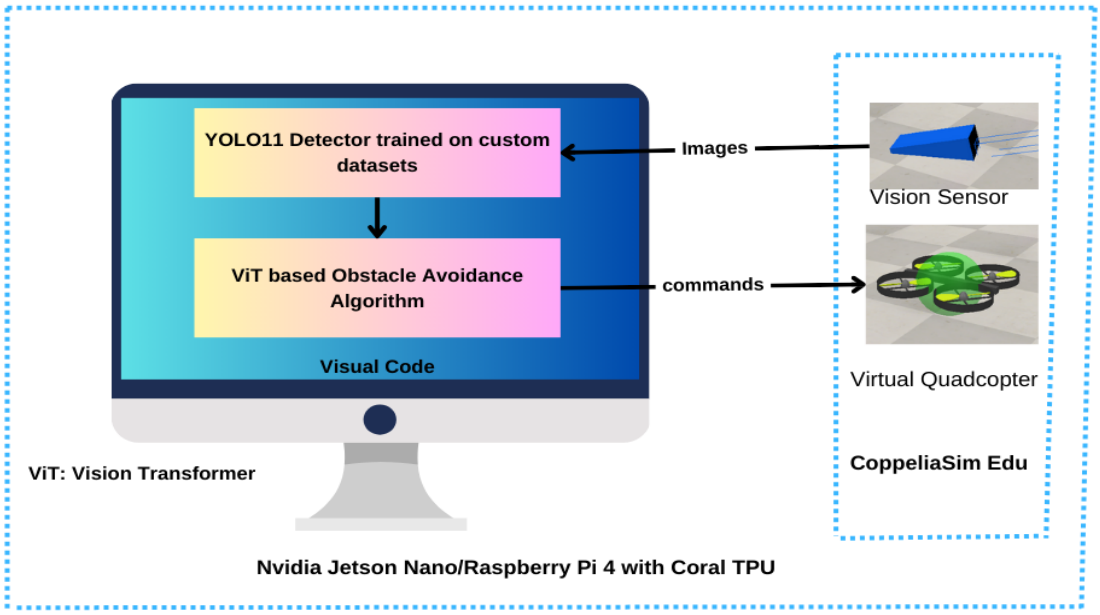

Figure 3.4. System architecture design

More sequential flow of data in the designed system is outlined in Table 3.2.

Component	Input	Role	Output
YOLO	Raw image	Detect objects	Bounding boxes + labels
ViT	Cropped object regions	Ccontextual importance	Encoded state
PPO	Combined ViT features	Decision making	Navigation action

Table 3.2. Each AI models usage

3.5. Simulation Environment

Simulation is pivotal for developing, training, and testing the UAV-based object detection and the obstacle avoidance systems before actual deployment. For the purpose of this research, the simulation platform chosen was CoppeliaSim (version 4.9) because of its powerful dynamics engine, sensor simulation in real-time, and robust support for the ZMQ RemoteAPI. The configuration made it possible to perform a high-fidelity simulation of aerial envrionments with the capacity for the closed-loop training as well as the evaluation of object detection and deep reinforcement learning-based control policies.

3.5.1. CoppeliaSim Platform

CoppeliaSim is a development platform specifically designed for simulating robots, with precise kinematics, dynamics, and sensor simulation. For this research, its vision sensor components were used to simulate the input from an RGB camera, and custom scripting allowed the streaming and control feedback of the image in real-time via the Python Remote API. Support from the platform for the dynamic pose of obstacles, realistic illumination, and the detection of collisions enabled precise detection algorithm testing under different conditions. It also allowed for the simple integration with learning frameworks by supporting frame-by-frame state extraction and external model inference (Montenegro et al., 2022)

3.5.2. UAV Model

One quadrotor UAV model was set up in CoppeliaSim with a simulated movable vision camera with gimbal yaw and gimbal pitches included for the Z direction movement and X direction movements respectively to mimic the real UAV's vision system. The UAV was controlled through Python API scripts that mimicked actual movement commands. The sensor was placed to resemble a regular onboard camera mount, oriented with the UAV's forward vector to ensure realistic aerial views. The model facilitated smooth testing of both vision-based object detection and action choosing by the PPO-based navigation agent.

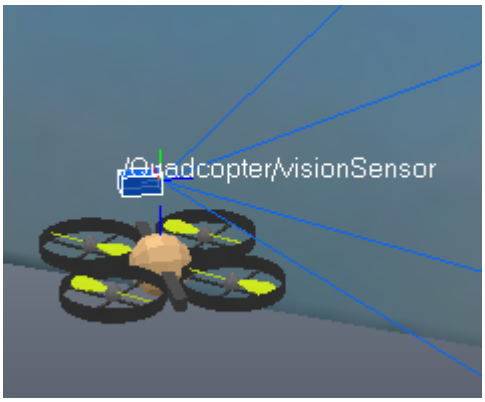

Figure 3.5. Quadcopter in CoppeliaSim

3.5.3. Environment Configuration

The simulation infrastructure was designed to be both dynamic and module-based in order to allow for the creation of different sets of obstacles. Birds, buildings, and trees were placed either static or with pre-set movement to simulate real obstacles like aerial obstacles in flight or changing city buildings. Environmental conditions were varied through randomized positions for the obstacles, altered camera positions, and different illumination conditions. The setting made the trained models robust and generalizable when transferred to real-time embedded deployment by exposing them to broad spatial and temporal complexities

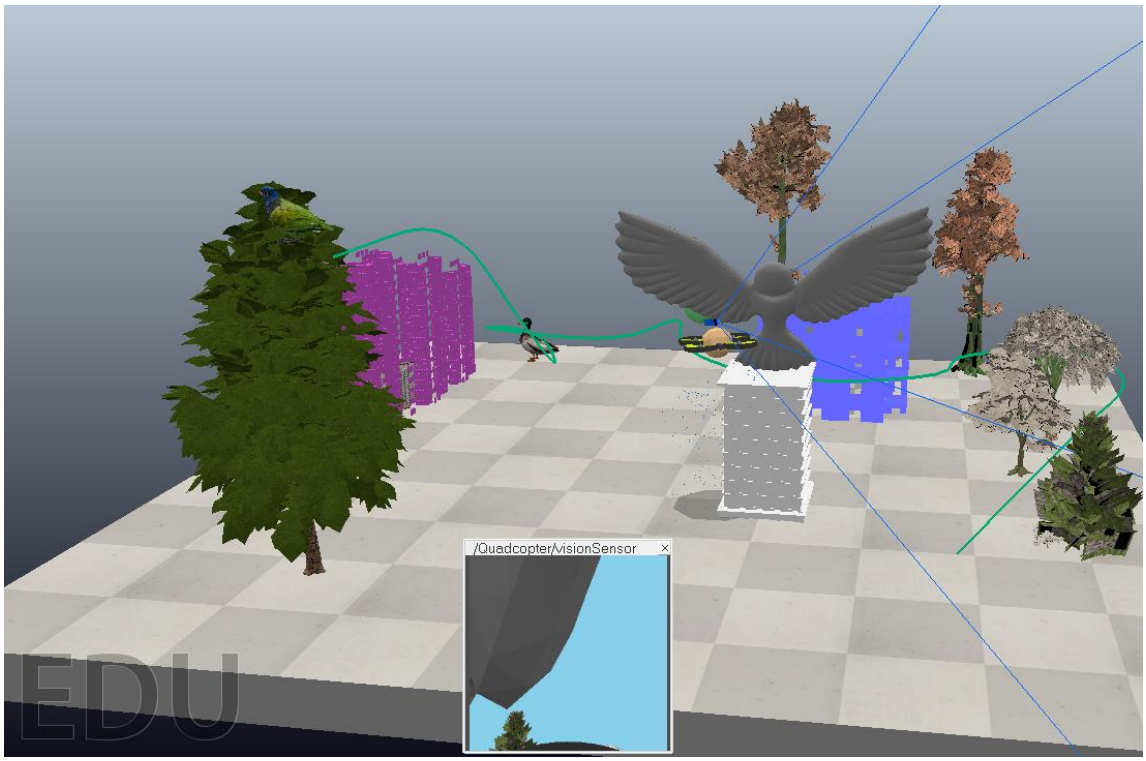

Figure 3.6(a). Simulation Environment

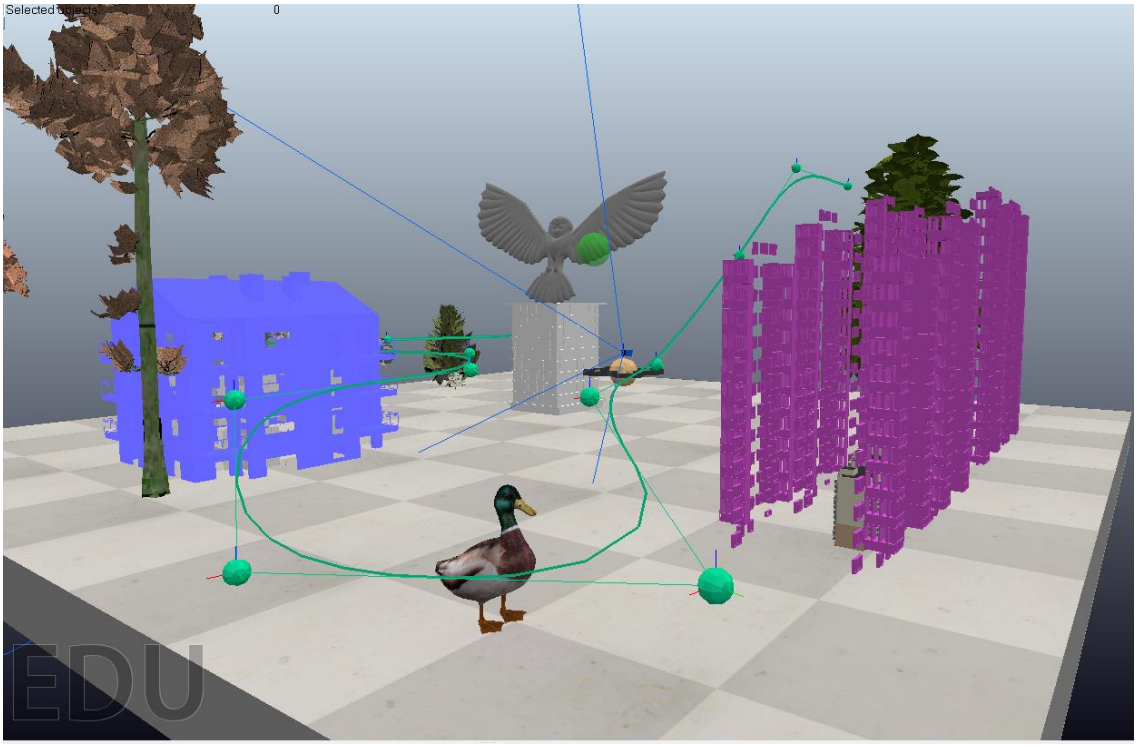

Figure 3.6(b). Simulation Environment

3.6. Training of Proposed Models

The training process included two main aspects: object detection model training with YOLOv11n and obstacle avoidance training with a Vision Transformer (ViT) combined with Proximal Policy Optimization (PPO). The two models were first trained in a simulated setting prior to deployment onto edge devices.

What was distinctive in this implementation is the fact that YOLOv11n output was used to create input to the PPO, in contrast to standard raw image input or hand-designed states. Combining such rich semantic visual features with transformer-based learning allowes the model to perceive its environment more comprehensively, particularly in cluttered and complex conditions. Although there are recent works that apply transformer-based RL in robotics, this is the first to benchmark an end-to-end UAV system using real embedded hardware, with direct linkage to object-aware decision-making.

3.6.1. Object Detection

The YOLOv11 object detection model was custom trained with a Roboflow dataset of annotated images of buildings, trees, and birds, pivotal classes for UAV navigation. The dataset was prepared and augmented with Roboflow using horizontal flipping, adjustments to the brightness, and rotation for the purpose of increasing the robustness of the model against changing flight conditions. The augmented dataset was then saved out in YOLO format to match the input requirements for the YOLOv11 model. Training was performed with both Colab GPU and local GPU with the Ultralytics framework.

Figure 3.7. Custom trained YOLO11n detection

3.6.2. Obstacle Avoidance

In the task of avoiding obstacles, a deep reinforcement learning methodology was used, combining a Vision Transformer (ViT) with the Proximal Policy Optimization (PPO) algorithm. The ViT transformed the video inputs from the onboard camera of the UAV into feature vectors that represent the semantic knowledge of the surroundings. The vectors formed the state space for the PPO agent, and it decided the best action to be undertaken for avoiding obstacles.

3.6.3. Probability in Deep Learning Models

Within the framework of PPO, the policy produces a probability distribution over the available actions so the agent can efficiently balance exploitation and exploration. The process of training included sampling trajectories from interaction with the environment, calculating advantage estimates, and policy updating by maximization of a clipped surrogate objective function. The reward function was carefully crafted to prompt the UAV to move efficiently to the destination avoiding collisions, with penalty for closeness to obstacles and divergence from the optimal trajectory(J. Wang et al., 2024).

A. **PPO (Proximal Policy Optimization) – Obstacle Avoidance**

1. Clipped Surrogate Objective Function

$$L^{CLIP}(\theta) = E_t\left[\min\left(r_t(\theta)\ddot{A}, clip(r_t), 1-\in, 1+\in\right)\ddot{A}_t\right] \quad (3.1)$$

This is the core PPO update rule. It tries to improve the current policy by comparing it with the previous policy using a probability ratio $r_t(\theta)$. The clip prevents drastic updates that could destabilize training. This ensures a stable learning process, which is critical for UAVs navigating dynamic obstacle fields(Schilling et al., 2021).

2. Advantage Estimation (GAE)

$$\hat{A} = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \quad (3.2)$$

$$where\ \delta_t = r_t + \gamma V(S_{t+1}) - V(s_t)$$

The **Advantage Function** estimates how much better a specific action is compared to the average expected value. This helps the agent **focus updates** only on actions that improved performance. In your UAV setup, it means learning to reinforce only those flight maneuvers that successfully avoided collisions(Schilling et al., 2021).

3. Entropy Bonus

$$L^{ENTROPY} = \beta \cdot E_t\left[H[\pi_\theta(\cdot|s_t]\right] \quad (3.3)$$

This term encourages the agent to keep its action distribution diverse, especially early in training. It **promotes exploration** by penalizing overly confident, premature decisions. For UAVs, this means exploring different avoidance paths to ensure the policy doesn't get stuck in suboptimal behaviors.

B. YOLOv11 – Object Detection

1. Objectness Loss

$$\mathcal{L}_{obj} = \sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\ (P_i - \hat{P}_i)^2 \quad (3.4)$$

This term measures how confidently the model predicts whether there's an object in a specific grid cell. A good objectness score is essential for the UAV to spot obstacles like trees, birds, or buildings in its visual field(Z. Wang et al., 2023).

2. Bounding Box Regression Loss

$$\mathcal{L}_{bbox} = \lambda_{coord} \sum[(x - \hat{x})^2 + (y - \hat{y})^2 + \left(\sqrt{\omega} - \sqrt{w}\right)^2 + (\sqrt{h} - \sqrt{\hat{h}})^2] \quad (3.5)$$

This loss calculates the difference between the predicted box position and the actual object's position. Precise bounding boxes are crucial for the drone to know exactly where an obstacle is and take the correct avoidance action.

3. Classification Loss

$$\mathcal{L}_{class} = \sum(p(c) - \hat{p}(c))^2 \quad (3.6)$$

This term ensures that the object (bird, tree, or building) is correctly classified. Misclassification could lead the UAV to wrongly prioritize or ignore obstacles.

4. Total Loss

$$\mathcal{L}_{total} = \mathcal{L}_{bbox} + \mathcal{L}_{obj} + \mathcal{L}_{class} \quad (3.7)$$

This combines all individual components and represents the final cost that YOLOv11 minimizes during training. A well-trained detection model provides clean, timely visual data to the PPO agent, enabling safer flight(Z. Wang et al., 2023).

3.7. Edge Device Configuration

This section delineates the hardware platforms utilized for the deployment of the trained object detection and obstacle avoidance models, emphasizing their specifications, machine learning pipelines, and optimization techniques.

3.7.1. NVIDIA Jetson Nano

The NVIDIA Jetson Nano J1010 is an edge AI computing platform for the deployment of machine learning models in real-time. It comes with a 128-core Maxwell GPU, a quad-core ARM Cortex-A57 CPU, 4GB of LPDDR4 RAM, and 16GB of eMMC. The configuration supports up to 0.5 TFLOPs of FP16 computing performance, which is ideal for efficiently running complex deep learning models. The product runs using the NVIDIA JetPack SDK with the CUDA, cuDNN, and TensorRT libraries, supporting smooth integration and optimization of AI models(Choe et al., 2023).

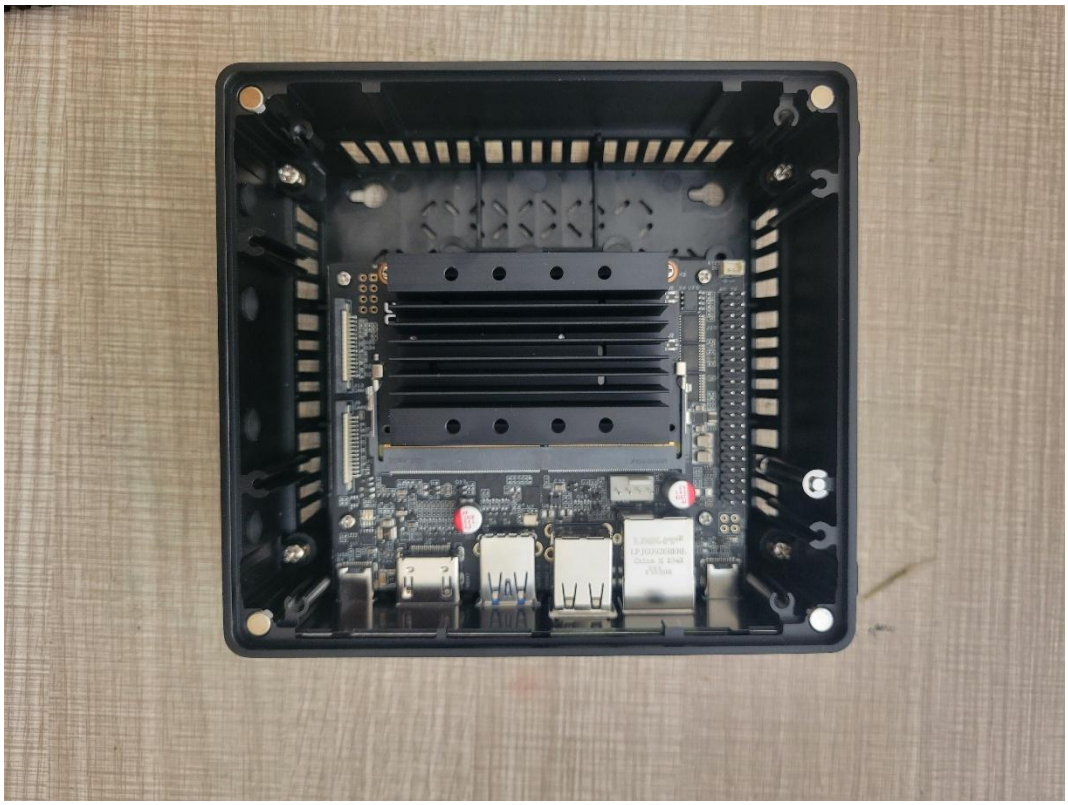

Figure 3.8. NVIDIA Jetson Nano J1010

The models are typically converted to the ONNX form and then optimized by using TensorRT for machine learning deployment, which accelerates the inference by harnessing the capability of the GPU. The optimization involves techniques such as layer fusion and precision calibration (e.g., FP16) such that the latency is decreased and the throughput is enhanced. All such optimizations are essential for UAV applications for which instantaneity is a must(Swaminathan et al., 2024).

3.7.2. Raspberry Pi 4 + Coral TPU

The Raspberry Pi 4 Model B with a quad-core ARM Cortex-A72 CPU and a maximum of 8GB LPDDR4 RAM is a low-cost platform for edge computing applications. To enhance its machine learning functionality, the Google Coral USB Accelerator with the Edge TPU coprocessor is included. The Edge TPU can handle 4 trillion operations per second (TOPS) with very little power consumption and is well-suited for running quantized neural networks(Dubois, n.d.).

Figure 3.9. Raspberry Pi 4 & Coral TPU

In this configuration, light-weight models like the ones based on the MobileNet architecture are quantized to 8-bit integers and saved to TensorFlow Lite form for compatibility with the Edge TPU. The configuration facilitates efficient low-latency inference adequate for the application

requirements of object detection and obstacle detection in UAV applications. The Raspberry Pi 4 coupled with Coral TPU achieves a balance between performance and power consumption, critical for aerial autonomous applications.

3.8. Performance Metrics

To assess the overall performance of the UAV platform for object detection and collision avoidance in real-time, four main performance measures were chosen. The measures not only test detection quality, but also system responsiveness and power consumption across various embedded boards.

3.8.1. Inference Time

Inference time represents the average time taken by the model to process a single input and produce an output. This metric is critical in UAV applications where delays in decision-making could lead to collisions or mission failure(Alqahtani et al., 2024).

Formula:

$$tinf = \frac{T_{end} - T_{start}}{\mathrm{N}} \quad (3.8)$$

Where:

T_{start} and T_{end} are the timestamps before and after inference

N is the number of frames processed

3.8.2. Energy Efficiency

This metric measures how much energy is consumed by the system to process each input frame. For battery-operated UAVs, maintaining low energy per inference is essential for extended flight time(Swaminathan et al., 2024).

$$E_{eff} = \frac{P_{total} - P_{idle}}{N} \quad (3.9)$$

Where:

P_{total} power consumption during active processing

P_{idle} baseline power consumption

N number of frames processed

Later Energy(J) is calculated by multiplying E_{eff} with the processing time.

3.8.3. Model Precision (mAP)

The mean Average Precision (mAP) measures how well the object detection model identifies and classifies objects across different categories. It combines both precision and recall across various confidence thresholds(Alqahtani et al., 2024).

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \quad (3.10)$$

Where:

AP_i average precision for class

N total number of object classes

3.8.4. Collision Rate

Collision rate evaluates the navigational success of the UAV, defined by how frequently the drone collides with obstacles during test episodes. This metric is vital for validating the reliability of the PPO-based obstacle avoidance system.

$$\text{Collision Rate} = \frac{\text{Number of Collisions}}{\text{Total Number of Trials}} \quad (3.11)$$

CHAPTER FOUR
RESULT AND DISCUSSION

This chapter describes the main results of the system's deployment and evaluation, with a focus on the performance of the UAV in real-time object detection and obstacle avoidance tasks leveraging two embedded boards, the NVIDIA Jetson Nano and the Raspberry Pi 4 with Coral TPU. It examines the performance metrics: inference time, power consumption, rate of collision, and reward accumulated.

4.1. Simulation setup and evaluation parameters

The simulation was setup using CoppeliaSim, a powerful physics based environment that enables the testing of UAV navigation algorithms under controlled conditions. The setup has a quadcopter as a representation of UAV which was modeled with idealized physical profile as well as weight tuned for stability and responsiveness within a bounded 3D space. The obstacles in the environment are stationary but responsive properties.

The UAV interacted with the obstacles through vision sensor and YOLOv1ln detection model followed by a ViT based PPO trained for obstacle avoidance. The simulation was orchestrated through ZMQ Remote API with low-latency communication between CoppeliaSim and the NVIDIA Jetson Nano or Raspberry Pi with TPU.

Looking more practically into the performance metrics, **inference time** per step refers to the total duration required for the UAV to complete one full cycle of sensing, detection, feature extraction and decision making. **Energy consumption** per step qualifies the amount of power consumed during this process, directly influencing battery life and mission duration. Another one is **reward performance** captures how well the agent is achieving obstacle avoidance where as collision rate reflects the safety and reliability of the avoidance. On that note, the introduction of ViT is expected to enhance navigation decisions but with the cost of more inference time and energy consumption.

4.2. Evaluation of the Custom trained YOLOv11n Model

Detection Accuracy Metrics

To measure the accuracy of object detection of the model that was customize trained YOLOv11, different evaluation metrics were in consideration. They comprise precision, recall, and mean Average Precision (mAP) over different Intersection over Union (IoU) ranges. The mAP, which is a common benchmark in object detection, provides a general indication of the ability of the model to detect and localize objects correctly.

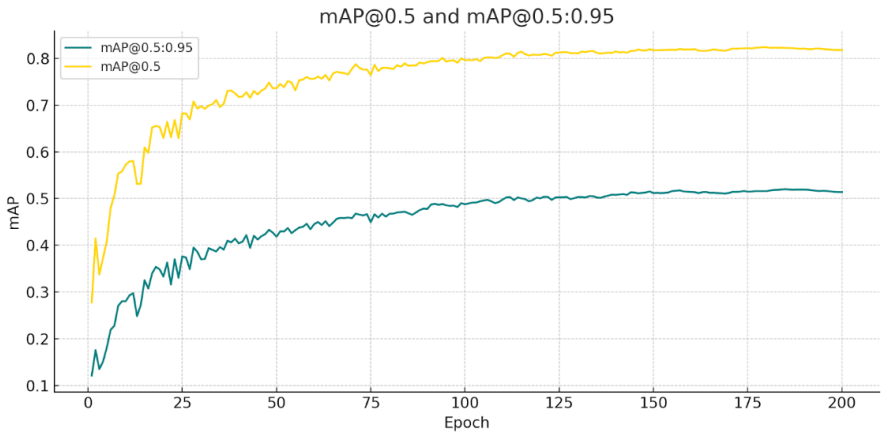

Figure 4.1. YOLOv11 mAP

As we can see, mAP@0.5 consistently reported higher values, which were indicative of the strength of the model with loose localization demands. mAP@0.5:0.95, however, offered a stricter testing, further indicating the strength of the model with different demands of detection.

Confusion Matrix and Class-wise Performance

The confusion matrix is not just a performance evaluation tool but also a diagnostic instrument that provides insights into specific areas where a classification model may need improvement, making it indispensable in the development and refinement of machine learning models(Zhao, 2025).

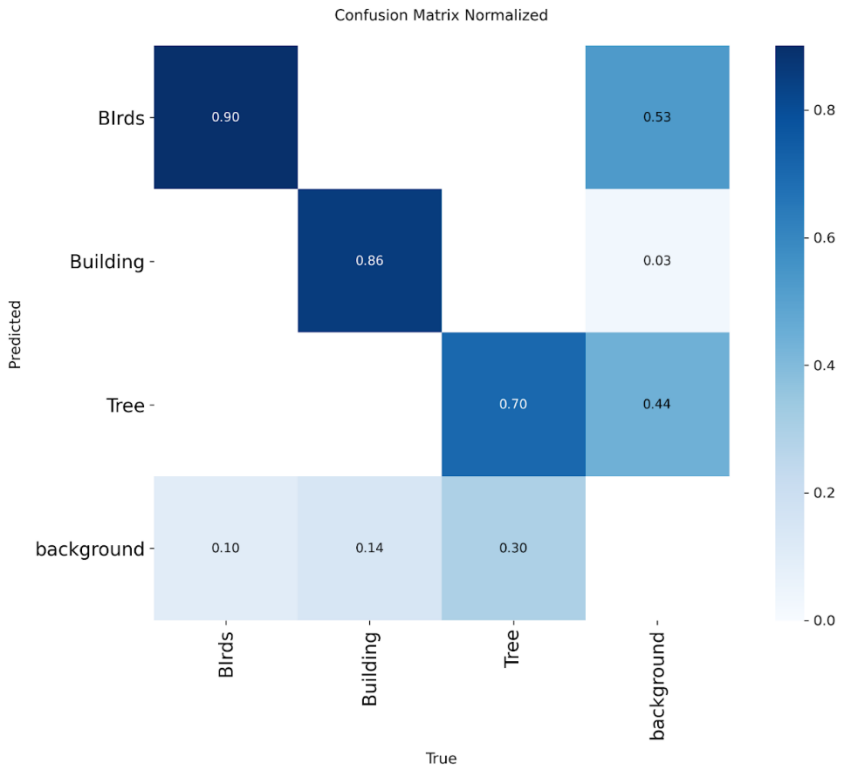

Figure 4.2. Normalized Confusion Matrix

The confusion matrix obtained for the dataset shows good detection and classification ability, especially in the discrimination of primary object classes such as birds, buildings, and trees. The majority of the forecasts closely match the true labels with 90%, 86% and 70% of the birds, buildings and trees being respectively and accurately identified. The misclassifications are minimal.

4.3. Performance Evaluation

4.3.1. Inference Time Analysis

A critical factor in real-time UAV navigation is the speed at which the onboard system can process visual data and make decisions. In this study, inference time is quantified as the total processing time to conduct end-to-end processing per decision cycle in the UAV simulation. This

entails capturing visual data from the CoppeliaSim-simulated camera, processing the same using the YOLOv11 model for object detection, extracting the features using the Vision Transformer (ViT), and applying the reinforcement learning (PPO) policy to establish the next action of the UAV. The timing incorporates the entire pipeline from perception to decision-making and reflects an overall measure of system responsiveness to real-time navigation tasks across the NVIDIA Jetson Nano (with TensorRT acceleration) and the Raspberry Pi 4 with Coral TPU (with EdgeTPU-supporting models).

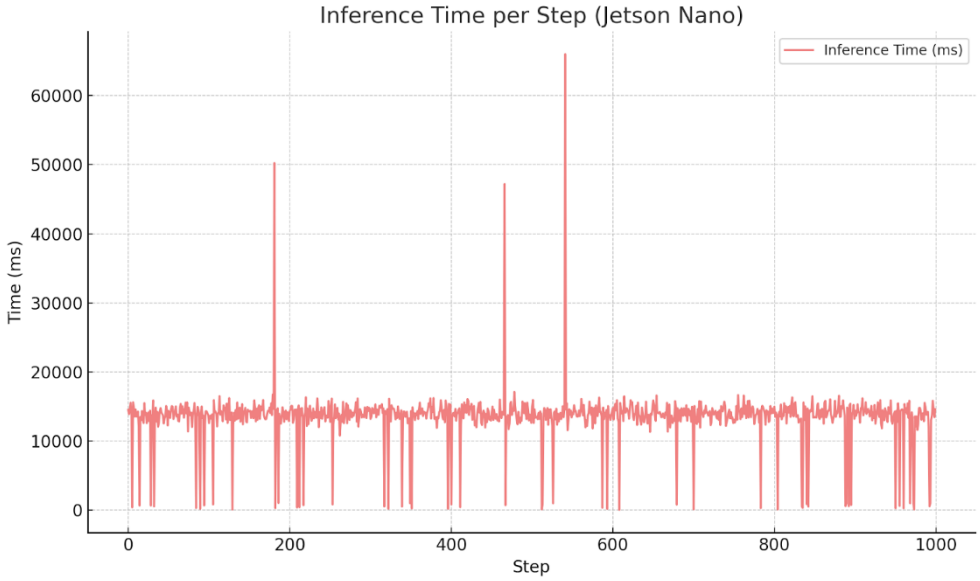

Figure 4.3. Inference time(NVIDIA)

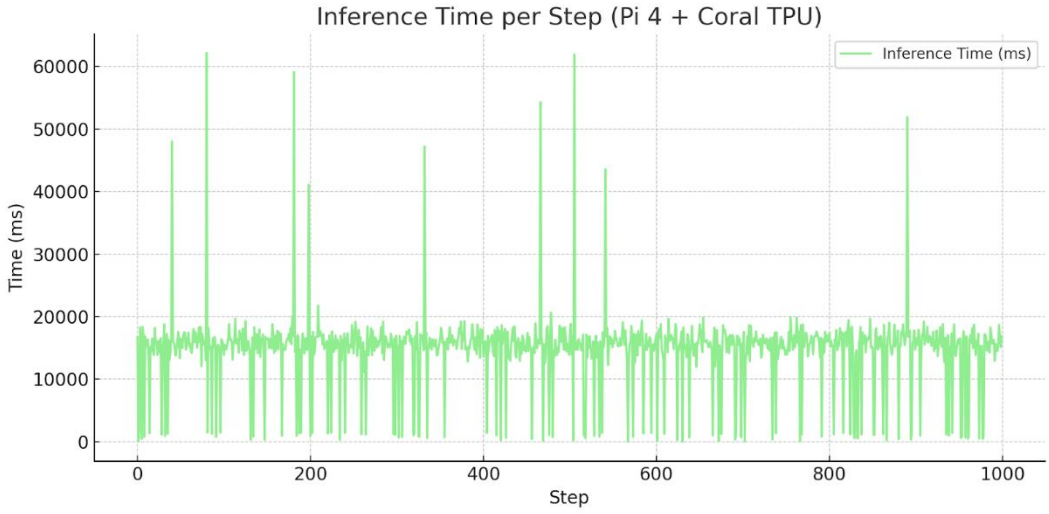

Figure 4.4. Inference time(Pi4 + TPU)

4.3.2. Energy Consumption

In this study, energy is approximated using software-accessible power interfaces found on the Jetson Nano and relied on assumptions on the Raspberry Pi 4 + Coral TPU. CPU and GPU power from power rails was read from the onboard file system for Jetson Nano and converted to joules per increment of inference time. For the TPU + Pi 4, active power was obtained from device specifications and converted to energy by multiplying with the measured processing time per frame. This is a non-intrusive and hardware-free way to estimate energy efficiency in real-time UAV running tasks.

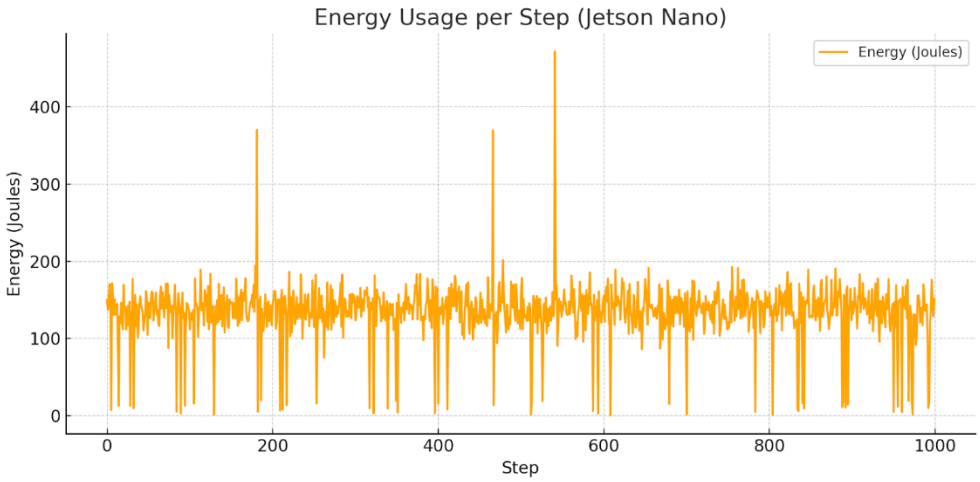

Figure 4.5. Energy Consumption(NVIDIA)

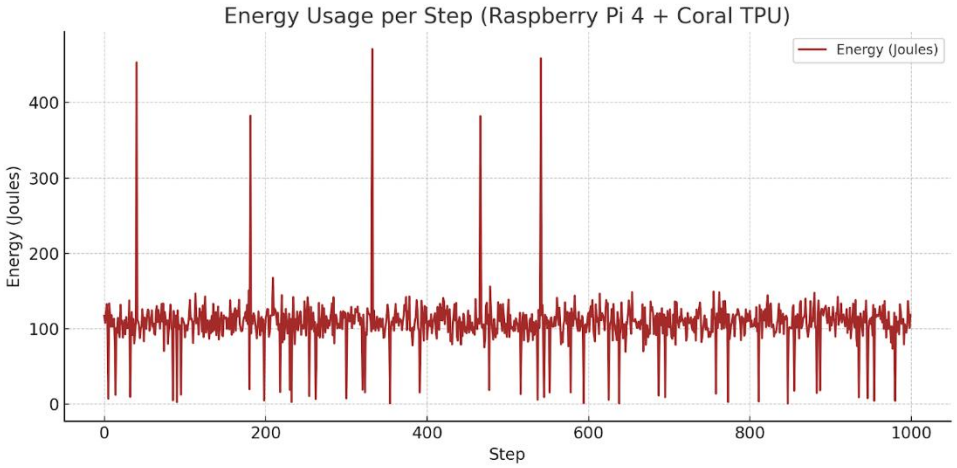

Figure 4.6. Energy Consumption(Pi4 &TPU)

4.3.3. Reward Performance

Reward performance is the aggregated or mean feedback signal obtained by the UAV agent in the form of how well the agent is reaching its navigation goal in the course of simulation. In this experiment, reward was calculated in each step using detection confidence, target distance, and penalties for collision and for exceeding height, as a quantitative measure of the decision quality of the agent throughout the control loop. The reward performance in the current implementation varied within a range of different navigation results, with other statistics including the minimum, the maximum, and the standard deviation also used to judge the consistency and efficiency of the policy. From this analysis, the agent is determined to see if, in every scenario, it is reaching its target goals or if there is a need to fine-tune its behavior to stabilize over different situations.

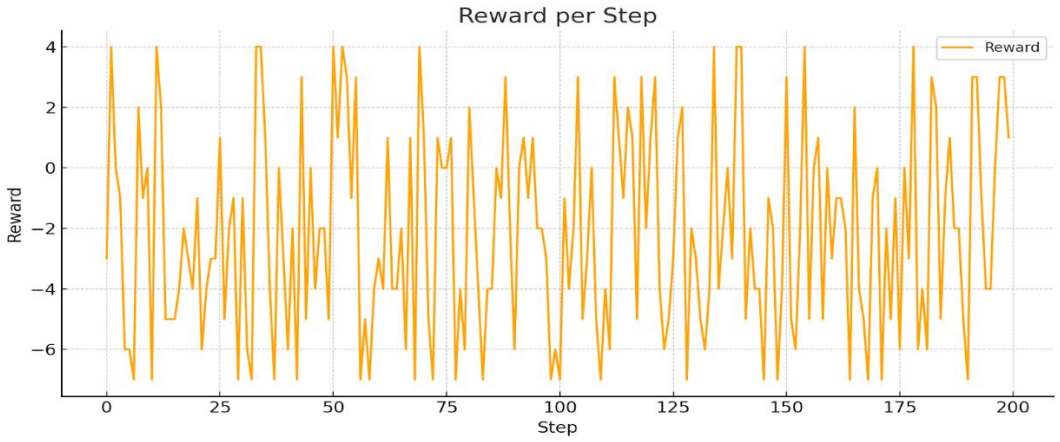

Figure 4.7. Reward Performance(NVIDIA)

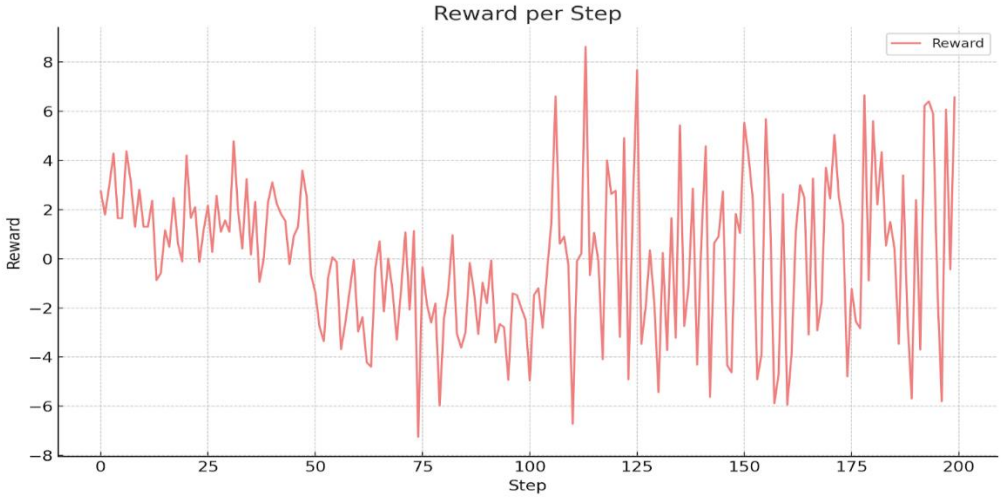

Figure 4.8. Reward Performance(Pi4 &TPU)

4.3.4. Collision Rate

The collision behavior was measured through the monitoring of the frequency of UAV encounters with the obstacle in the simulation. Decreased collision frequency represented good obstacle avoidance by the PPO agent as a result of the visual context from the ViT embeddings and stable decision making across different environmental conditions.

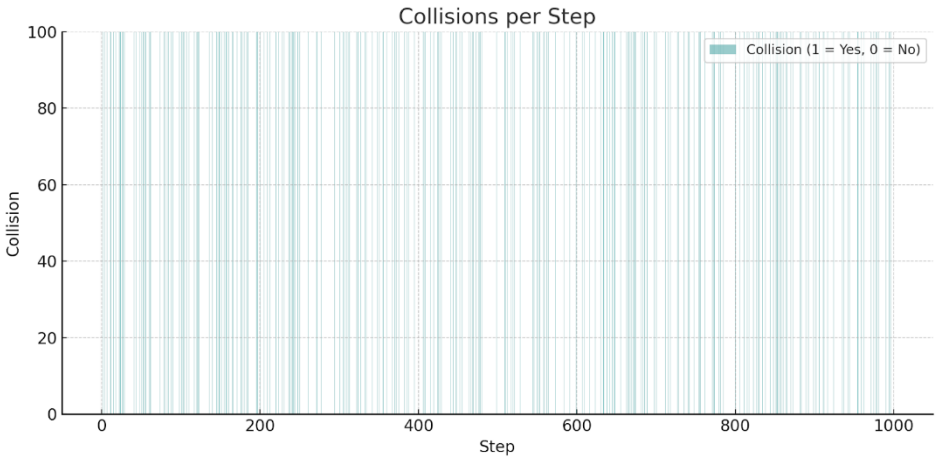

Figure 4.9. Collision Rate(NVIDIA)

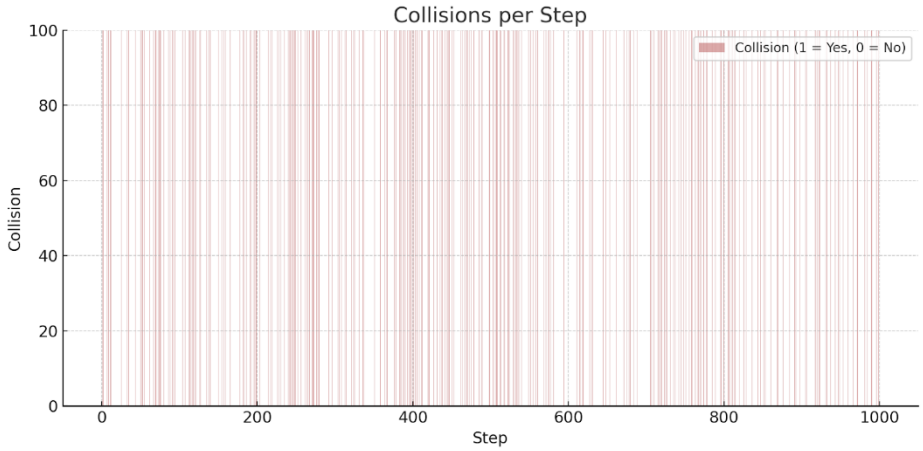

Figure 4.10. Collision Rate(Pi4 & TPU)

4.3.5. Comparative Benchmarking

4.3.5.1. Benchmarking current metrics

Deploying the models on Raspberry Pi 4 with Coral TPU provided a significantly more complex than on the NVIDIA Jetson Nano, though both platforms came with their own set of challenges. The Jetson Nano's setup was constrained by outdated Python version and JetPack, which limited access to newer libraries and required workaround implementations. On the hand, the TPU demanded a very specific workflow, supporting only TFLite models with quantizes formats. As a matter of fact, integrating complex models like ViT and PPO was notably restrictive due to resource limitations and compatibility hurdles. These constraints required compromises in model depth. Despite the difficulties, the TPU achieved commendable energy efficiency even though it underperformed in inference time, reward, and collision rate.

Performance Metrics	Jetson Nano	Pi 4 + TPU
Inference time	14031ms	16104ms
Energy Consumption	167.32 J	138.09 J
Reward Performance	-7.72	-10.73
Collision Rate	25%	30%

Table 4.1. Results comparison

4.3.5.2. Benchmarking with previous study

According to previous studies, full end-to-end object detection and obstacle avoidance systems are not present that are implemented on edged devices. However, the benchmark can be made with systems that only focused on navigation or obstacle avoidance like(Calderon-Cordova & Sarango, 2023) which introduces a Deep Reinforcement Learning (DRL) algorithm for robotic manipulation and navigation in simulated environments, tested on an NVIDIA GeForce. Their work, "A Deep Reinforcement Learning Algorithm for Robotic Manipulation Tasks in Simulated Environments," achieves a reward performance of -1.1 to -0.6 per step, notably better than this

study's -7.72 (Jetson Nano) and -10.73 (Pi 4 + TPU), likely due to their simpler DRL approach without the complexity of YOLOv11 or ViT, and a less demanding simulation compared to CoppeliaSim's realistic physics. However, this study enhances by integrating YOLOv11 and ViT-PPO for a complete detection-to-navigation pipeline on edge devices, even if the complex models bump up resource use.

4.4. Discussion and Key Observations

The comparison of the Jetson Nano with the Raspberry Pi 4 + TPU indicates a balance between processing speed, efficiency, and real-time decision making in UAV navigability. The Jetson Nano achieved a much lower mean inference time of 14.03 seconds per frame to 16.10 seconds for the Pi 4 + TPU. The reason is probably GPU acceleration-based as well as TensorRT optimization, which allows the Jetson to process input from the vision more rapidly and consistently. Both the systems were well within acceptable real-time limits, but the faster inference of the Jetson can make a big impact in handling dynamic scenes with moving objects or sudden changes in input.

From the control quality aspect, the Jetson also performed slightly better in decision-making. It achieved a greater average reward of -7.72 versus -10.73 and observed lower collision rates (25% versus 30%), indicating that the PPO agent executing on the Jetson learned more effective and stable control policies. But at the cost of this, this performance averaged 167.32 Joules per step versus 138.09 Joules on the Pi 4 + TPU. It can thus be inferred that the Pi 4 + TPU is comparatively more power-efficient but loses in terms of responsiveness and stability to some extent. All in all, in applications where responsiveness and robustness of the system is of prime importance, the Jetson Nano can prove to be more suitable despite its increased power usage.

On another note, recent developments in UAV navigation have increasingly ventured into the methodologies of using transformer-based architectures in tandem with reinforcement learning to augment autonomous decision-making. For instance, the DTPPO framework used a Dual-Transformer Encoder in the context of a PPO algorithm to bring better generalization and obstacle avoidance in unpredictable environments(Wei et al., 2024). By comparison, the ViT + PPO

combination in the work presented here is unique in its design, harnessing the output of real-time object recognition from YOLOv11 as input features. The process translates the higher-level visual cues to useful states for decision-making, essentially marrying semantic perception from Vision Transformers with reinforcement learning. Built in the context of a simulated UAV scenario, the end-to-end model exhibited good starting performance, indicating its potential as a real-world and resilient solution for intelligent UAV navigation. And the result found here is just the beginning as there is a long way to improvement with different reward systems and more trainings for the RL model with dedicated steps will drastically change the performance metrics.

CHAPTER FIVE
CONCLUSION AND RECOMMENDATIONS

5.1. Conclusion

The results of this study show that Jetson Nano had the edge in inference time, owing to its optimized TensorRT integration and GPU-based execution. Even though the Pi 4 with TPU had lower energy consumption, its slower processing and higher collision rates showed that energy efficiency alone doesn't ensure robust performance in navigation tasks. The Jetson Nano was proven to be capable of handling complex model combination implementation by showing more reliable results in terms of response time, control quality, and stability. Moreover, the ViT-based PPO pipeline effectively bridged the gap between perception and control, proving the potential of combining deep vision models with reinforcement learning for real-world robotic systems. These results provide a solid starting point for future research in edge AI for drones.

5.2. Recommendations

The intention of this thesis was to benchmark and compare the real-time performance of object detection and customer modified obstacle avoidance systems across two embedded platforms. The focus remained on evaluating how well the system components, particularly the YOLOv11 and ViT-integrated PPO models, performed in simulated UAV environments, rather than on optimizing every element of the implementation.

For future work, it is recommended to further refine the following points.

- Extend the simulation-based benchmarking to full hardware implemented benchmarking.
- Further refine the integration between the Vision Transformer and PPO architecture to enhance decision responsiveness, reduce collision occurrences, and maintain consistent reward trajectories.
- Consider implementing the model on more capable hardware platforms, such as NVIDIA

Jetson Xavier NX or Orin, to reduce inference latency and improve real-time performance.

- Evaluate deployment on other edge AI devices like Google EdgeBox or Intel Movidius for broader benchmarking and energy-performance balance.
- Extend the framework beyond UAVs to other autonomous systems such as ground robots or mobile manipulators where vision-based decision making is crucial.

REFERENCES

Alam, M. S., Natesha B. V., Ashwin T. S., & Guddeti, R. M. R. (2019). UAV based cost-effective real-time abnormal event detection using edge computing. *Multimedia Tools and Applications*, *78*(24), 35119–35134. https://doi.org/10.1007/s11042-019-08067-1

Albanese, A., Nardello, M., & Brunelli, D. (2022). Low-power deep learning edge computing platform for resource constrained lightweight compact UAVs. *Sustainable Computing: Informatics and Systems*, *34*, 100725. https://doi.org/10.1016/j.suscom.2022.100725

Alqahtani, D. K., Cheema, A., & Toosi, A. N. (2024). *Benchmarking Deep Learning Models for Object Detection on Edge Computing Devices*. http://arxiv.org/abs/2409.16808

Baller, S. P., Jindal, A., Chadha, M., & Gerndt, M. (2021). *DeepEdgeBench: Benchmarking Deep Neural Networks on Edge Devices*. http://arxiv.org/abs/2108.09457

Bemposta Rosende, S., Ghisler, S., Fernández-Andrés, J., & Sánchez-Soriano, J. (2023). Implementation of an Edge-Computing Vision System on Reduced-Board Computers Embedded in UAVs for Intelligent Traffic Management. *Drones*, *7*(11), 682. https://doi.org/10.3390/drones7110682

Benjaminlapid. (2023, August 17). *https://www.slideshare.net/slideshow/how-is-a-vision-transformer-vit-model-built-and-implemented/259970300*.

Calderon-Cordova, C., & Sarango, R. (2023). A Deep Reinforcement Learning Algorithm for Robotic Manipulation Tasks in Simulated Environments †. *Engineering Proceedings*, *47*(1). https://doi.org/10.3390/engproc2023047012

Chen, C., Liu, S. D., & Zhang, S. (2024). *Efficient Policy Evaluation with Safety Constraint for Reinforcement Learning*. http://arxiv.org/abs/2410.05655

Chen, C., Zheng, Z., Xu, T., Guo, S., Feng, S., Yao, W., & Lan, Y. (2023). YOLO-Based UAV Technology: A Review of the Research and Its Applications. In *Drones* (Vol. 7, Issue 3). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/drones7030190

Choe, C., Choe, M., & Jung, S. (2023). Run Your 3D Object Detector on NVIDIA Jetson Platforms:A Benchmark Analysis †. *Sensors*, *23*(8). https://doi.org/10.3390/s23084005

Dubois, E. (n.d.). *Calhoun: The NPS Institutional Archive DSpace Repository SHARED LEARNING AMONG DISTRIBUTED EDGE DEVICES USING CORAL EDGE TPU MACHINE LEARNING ENGINES*. https://hdl.handle.net/10945/67703

Fu, Z. (2022). *Vision Transformer: Vit and its Derivatives*. http://arxiv.org/abs/2205.11239

Gao, J., Liu, N., Li, H., Li, Z., Xie, C., & Gou, Y. (2025). Reinforcement Learning Decision-Making for Autonomous Vehicles Based on Semantic Segmentation. *Applied Sciences*, *15*(3), 1323. https://doi.org/10.3390/app15031323

Garcia-Perez, A., Miñón, R., Torre-Bastida, A. I., & Zulueta-Guerrero, E. (2023). Analysing Edge Computing Devices for the Deployment of Embedded AI. *Sensors*, *23*(23), 9495. https://doi.org/10.3390/s23239495

Ghasemi, M., Moosavi, A. H., & Ebrahimi, D. (2024). *A Comprehensive Survey of Reinforcement Learning: From Algorithms to Practical Challenges*. http://arxiv.org/abs/2411.18892

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., & Levine, S. (2018). *Soft Actor-Critic Algorithms and Applications*. http://arxiv.org/abs/1812.05905

Hakani, R., & Rawat, A. (2024). Edge Computing-Driven Real-Time Drone Detection Using YOLOv9 and NVIDIA Jetson Nano. *Drones*, *8*(11), 680. https://doi.org/10.3390/drones8110680

Han, D., Mulyana, B., Stankovic, V., & Cheng, S. (2023). A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation. *Sensors*, *23*(7), 3762. https://doi.org/10.3390/s23073762

Hsieh, T.-L., Jhan, Z.-S., Yeh, N.-J., Chen, C.-Y., & Chuang, C.-T. (2023). An Unmanned Aerial Vehicle Indoor Low-Computation Navigation Method Based on Vision and Deep Learning. *Sensors*, *24*(1), 190. https://doi.org/10.3390/s24010190

Jabr, I., Salman, Y., Shqair, M., & Hawash, A. (2025). Penetration Testing and Attack Automation Simulation: Deep Reinforcement Learning Approach. *An-Najah University Journal for Research - A (Natural Sciences)*, *39*(1), 7–14. https://doi.org/10.35552/anujr.a.39.1.2231

Jayanth, R., Gupta, N., & Prasanna, V. (2024). *Benchmarking Edge AI Platforms for High-Performance ML Inference*. http://arxiv.org/abs/2409.14803

Jetson Nano Developer Kit. (2019).

Jiang, C., Ren, H., Ye, X., Zhu, J., Zeng, H., Nan, Y., Sun, M., Ren, X., & Huo, H. (2022). Object detection from UAV thermal infrared images and videos using YOLO models. *International Journal of Applied Earth Observation and Geoinformation*, *112*. https://doi.org/10.1016/j.jag.2022.102912

Karsan Marco Giordani, N., Asst Filippo Campagnaro, C., & Francescon, R. (n.d.). *Performance Analysis and Evaluation of Object Detection Algorithms for Drone Networks*.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in Vision: A Survey. *ACM Computing Surveys*, *54*(10s), 1–41. https://doi.org/10.1145/3505244

Khanam, R., & Hussain, M. (2024). *YOLOv11: An Overview of the Key Architectural Enhancements*. http://arxiv.org/abs/2410.17725

Khoi, N. D. H., Pham Van, C., Tran, H. V., & Truong, C. D. (2021). Multi-Objective Exploration for Proximal Policy Optimization. *Proceedings of 2020 Applying New Technology in Green Buildings, ATiGB 2020*, 105–109. https://doi.org/10.1109/ATiGB50996.2021.9423319

KIRAC, E., & ÖZBEK, S. (2024). Deep Learning Based Object Detection with Unmanned Aerial Vehicle Equipped with Embedded System. *Journal of Aviation*, *8*(1), 15–25. https://doi.org/10.30518/jav.1356997

Liu, M., Wang, X., Zhou, A., Fu, X., Ma, Y., & Piao, C. (2020). Uav-yolo: Small object detection on unmanned aerial vehicle perspective. *Sensors (Switzerland)*, *20*(8). https://doi.org/10.3390/s20082238

Ma, M.-Y., Shen, S.-E., & Huang, Y.-C. (2023). Enhancing UAV Visual Landing Recognition with YOLO's Object Detection by Onboard Edge Computing. *Sensors*, *23*(21), 8999. https://doi.org/10.3390/s23218999

MASF-YOLO: An Improved YOLOv11 Network for Small Object Detection on Drone View. (n.d.).

Mathe, S. E., Kondaveeti, H. K., Vappangi, S., Vanambathina, S. D., & Kumaravelu, N. K. (2024). A comprehensive review on applications of Raspberry Pi. *Computer Science Review*, *52*, 100636. https://doi.org/10.1016/j.cosrev.2024.100636

Meng, W., Zheng, Q., Pan, G., & Yin, Y. (2023). Off-Policy Proximal Policy Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(8), 9162–9170. https://doi.org/10.1609/aaai.v37i8.26099

Merei, A., Mcheick, H., Ghaddar, A., & Rebaine, D. (2025). A Survey on Obstacle Detection and Avoidance Methods for UAVs. In *Drones* (Vol. 9, Issue 3). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/drones9030203

Minott, D., Siddiqui, S., & Haddad, R. J. (2025). Benchmarking Edge AI Platforms: Performance Analysis of NVIDIA Jetson and Raspberry Pi 5 with Coral TPU. *SoutheastCon 2025*, 1384–1389. https://doi.org/10.1109/SoutheastCon56624.2025.10971592

Montenegro, G., Chacón, R., Fabregas, E., Garcia, G., Schröder, K., Marroquín, A., Dormido-Canto, S., & Farias, G. (2022). Modeling and Control of a Spherical Robot in the CoppeliaSim Simulator. *Sensors*, *22*(16), 6020. https://doi.org/10.3390/s22166020

Neil de la Fuente, & Daniel A. Vidal Guerra. (2024). A COMPARATIVE STUDY OF DEEP REINFORCEMENT LEARNING MODELS: DQN VS PPO VS A2C. *IEEE International Conference on Program Comprehension*, *2022-March*, 36–47. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Ortega, L. D., Loyaga, E. S., Cruz, P. J., Lema, H. P., Abad, J., & Valencia, E. A. (2023). Low-Cost Computer-Vision-Based Embedded Systems for UAVs. *Robotics*, *12*(6), 145. https://doi.org/10.3390/robotics12060145

Rachit Mehul Pathak, & Ajay Varma Mudunuri. (2023). Autonomous Obstacle Detection and Avoidance in Drones. *Innovative Research Thoughts*, *9*(4), 73–85.

https://doi.org/10.36676/irt.2023-v9i4-011

Romero, A., Aljalbout, E., Song, Y., & Scaramuzza, D. (2023). *Actor-Critic Model Predictive Control: Differentiable Optimization meets Reinforcement Learning*. http://arxiv.org/abs/2306.09852

Sapkota, R., Qureshi, R., Calero, M. F., Badjugar, C., Nepal, U., Poulose, A., Zeno, P., Vaddevolu, U. B. P., Khan, S., Shoman, M., Yan, H., & Karkee, M. (2025). *YOLOv11 to Its Genesis: A Decadal and Comprehensive Review of The You Only Look Once (YOLO) Series*. https://doi.org/10.1007/s10462-025-11253-3

Schilling, M., Melnik, A., Ohl, F. W., Ritter, H. J., & Hammer, B. (2021). Decentralized control and local information for robust and adaptive decentralized Deep Reinforcement Learning. *Neural Networks*, *144*, 699–725. https://doi.org/10.1016/j.neunet.2021.09.017

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. http://arxiv.org/abs/1707.06347

Singla, A., Padakandla, S., & Bhatnagar, S. (2021). Memory-Based Deep Reinforcement Learning for Obstacle Avoidance in UAV With Limited Environment Knowledge. *IEEE Transactions on Intelligent Transportation Systems*, *22*(1), 107–118. https://doi.org/10.1109/TITS.2019.2954952

Skarka, W., & Ashfaq, R. (2024). Hybrid Machine Learning and Reinforcement Learning Framework for Adaptive UAV Obstacle Avoidance. *Aerospace*, *11*(11), 870. https://doi.org/10.3390/aerospace11110870

Swaminathan, T. P., Silver, C., & Akilan, T. (2024). *Benchmarking Deep Learning Models on NVIDIA Jetson Nano for Real-Time Systems: An Empirical Investigation*. http://arxiv.org/abs/2406.17749

Tao, T., Reda, D., & van de Panne, M. (2022). *Evaluating Vision Transformer Methods for Deep Reinforcement Learning from Pixels*. http://arxiv.org/abs/2204.04905

Telli, K., Kraa, O., Himeur, Y., Ouamane, A., Boumehraz, M., Atalla, S., & Mansoor, W. (2023). A Comprehensive Review of Recent Research Trends on Unmanned Aerial Vehicles (UAVs). In *Systems* (Vol. 11, Issue 8). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/systems11080400

Theses, E., Theses, D., Papers, M., & Ravi Goteti, M. (2021). *Scholarship at UWindsor Scholarship at UWindsor Landmark Identification Using GPU for Autonomous Unmanned Landmark Identification Using GPU for Autonomous Unmanned Aerial Vehicle in GPS Denied Navigation Aerial Vehicle in GPS Denied Navigation*. https://scholar.uwindsor.ca/etd/8711

Toma, C., Popa, M., Iancu, B., Doinea, M., Pascu, A., & Ioan-Dutescu, F. (2022). Edge Machine Learning for the Automated Decision and Visual Computing of the Robots, IoT Embedded Devices or UAV-Drones. *Electronics*, *11*(21), 3507. https://doi.org/10.3390/electronics11213507

Vasishta, M. S. S., Reddy Amireddy, A. T., Shrivastava, P., S, R., Talasila, V., & Shastry, P. N. (2024).

Small Object Detection for UAVs Using Deep Learning Models on Edge Computing: A Comparative Analysis. *2024 5th International Conference on Circuits, Control, Communication and Computing (I4C)*, 106–112. https://doi.org/10.1109/I4C62240.2024.10748432

Vermesan, O., Diaz Nava, M., & Debaillie, B. (n.d.). *Industrial Artificial Intelligence Technologies and Applications*. www.riverpublishers.com

Wang, C., Han, Y., Yang, C., Wu, M., Chen, Z., Yun, L., & Jin, X. (2025). CF-YOLO for small target detection in drone imagery based on YOLOv11 algorithm. *Scientific Reports*, *15*(1), 16741. https://doi.org/10.1038/s41598-025-99634-0

Wang, J., Yu, Z., Zhou, D., Shi, J., & Deng, R. (2024). *Vision-Based Deep Reinforcement Learning of UAV Autonomous Navigation Using Privileged Information*. http://arxiv.org/abs/2412.06313

Wang, T., Ma, Z., Yang, T., & Zou, S. (2023). PETNet: A YOLO-based prior enhanced transformer network for aerial image detection. *Neurocomputing*, *547*, 126384. https://doi.org/10.1016/j.neucom.2023.126384

Wang, Y., He, H., & Tan, X. (n.d.). *Truly Proximal Policy Optimization*. https://github.com/

Wang, Y., He, H., Tan, X., & Gan, Y. (2019). *Trust Region-Guided Proximal Policy Optimization*. http://arxiv.org/abs/1901.10314

Wang, Z., Lei, L., & Shi, P. (2023). Smoking behavior detection algorithm based on YOLOv8-MNC. *Frontiers in Computational Neuroscience*, *17*. https://doi.org/10.3389/fncom.2023.1243779

Wei, A., Liang, J., Lin, K., Li, Z., & Zhao, R. (2024). DTPPO: Dual-Transformer Encoder-Based Proximal Policy Optimization for Multi-UAV Navigation in Unseen Complex Environments. *Drones*, *8*(12), 720. https://doi.org/10.3390/drones8120720

Wu, Q., & Zhou, Y. (2019). Real-Time Object Detection Based on Unmanned Aerial Vehicle. *2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS)*, 574–579. https://doi.org/10.1109/DDCLS.2019.8908984

Xue, C., Xia, Y., Wu, M., Chen, Z., Cheng, F., & Yun, L. (2024). EL-YOLO: An efficient and lightweight low-altitude aerial objects detector for onboard applications. *Expert Systems with Applications*, *256*, 124848. https://doi.org/10.1016/j.eswa.2024.124848

Yanes Luis, S., Gutiérrez-Reina, D., & Toral Marín, S. (2021). A Dimensional Comparison between Evolutionary Algorithm and Deep Reinforcement Learning Methodologies for Autonomous Surface Vehicles with Water Quality Sensors. *Sensors*, *21*(8), 2862. https://doi.org/10.3390/s21082862

Yang, X., & Wang, J. (2025). A Novel Reinforcement Learning Algorithm-Based Control Strategy for Grid-Configured Inverters. *Energies*, *18*(3), 597. https://doi.org/10.3390/en18030597

YOLO Explained: From v1 to Present. (n.d.).

Zhang, Q., Yan, F., Song, W., Wang, R., & Li, G. (2023). Automatic Obstacle Detection Method for the Train Based on Deep Learning. *Sustainability, 15*(2), 1184. https://doi.org/10.3390/su15021184

Zhao, N. , B. T. , Y. J. Y. , & D. K. (2025). *Outperformance score: a universal standardization method for Confusion-Matrix-Based classification performance metrics*.

APPENDIX

YOLOv11 Object Detection Training

1. Initialize YOLOv11 model with predefined architecture
2. Load pre-trained weights (optional)
3. For each training epoch:
   a. Load a batch of training images and corresponding labels
   b. Perform forward pass through the model
   c. Compute loss:
      - Object classification loss
      - Bounding box regression loss
      - Confidence score loss
   d. Backpropagate the loss to update model weights
   e. Validate on held-out data to monitor performance
4. Save the best-performing model weights

CoppeliaSim Environment Setup

1. Initialize environment:
   - Connect to CoppeliaSim using ZMQ Remote API
   - Load simulation scene and enable synchronous stepping
   - Load YOLO object detection models for different classes
   - Initialize ViT model for feature extraction (pretrained, frozen)
   - Define observation space (ViT features + UAV state + detection context)
   - Define action space for 3D movement control

2.Define _get_observation():
   - Capture vision sensor image and preprocess
   - Run YOLO detection and select top 3 confident boxes
   - For each box:

- Extract and preprocess cropped image region
- Use ViT to obtain feature vector

- Form observation by concatenating:
  - ViT features (1152 dims)
  - UAV position and velocity (6 dims)
  - Box context for top-3 objects (27 dims)

3.Define step(action):
- Update target position based on action and sinusoidal path
- Apply gravity compensation and constrain within simulation bounds
- Step the simulation and capture new observation
- Check for collisions and apply corresponding penalty
- Calculate distance-based reward and YOLO detection confidence penalty
- Return next observation, reward, termination flag, and info

4.Define reset():
- Stop and restart simulation
- Randomize UAV starting position within bounds
- Reset object positions and orientations
- Wait for stabilization, then return initial observation

5.Define _check_collision():
- Run YOLO on latest image
- Identify any large/confident objects near center as collision

6.Define render():
- Display current image frame with bounding boxes and labels

7.Define _compute_reward():
- Penalize high detection confidence (to encourage avoidance)
- Reward proximity to target based on Euclidean distance

PPO Training with Vision Transformer Features

Input: Environment with ViT-based observation, PPO configuration

Output: Trained PPO policy

1. Initialize PPO agent with MLP policy network
2. Connect PPO with custom UAV environment
3. For total_timesteps:
   a. Observe the current state from environment (ViT features)
   b. Select action using PPO policy (deterministic or stochastic)
   c. Execute action in the environment
   d. Collect reward, next state, and done signal
   e. Store experience in buffer
4. After n_steps:
   a. Compute advantage estimates and returns
   b. Update policy and value networks using collected rollouts
   c. Log reward and loss metrics
5. Save the final trained PPO model

Research Fund Acknowledgement

This research is funded by Adama Science and Technology University under the grant number ASTU/SM-R/1129/25, Adama, Ethiopia.