Reinforcement learning for object recognition and room classification in an indoor environment

Beatrice van Eden[1*] and *Natasha* Botha[2]

[1]Next Generation Enterprises and Institution, Future Artificial Intelligence and Extended Reality (AI&XR), CSIR, South Africa
[2]Centre for Robotics and Future Production, Council for Scientific and Industrial Engineering, Pretoria, South Africa

Abstract. This paper presents an integrated system for autonomous navigation and object prioritisation using a mobile robot in an indoor environment. The system combines deep learning for room classification (VGG16) and object detection (YOLOv8) with Proximal Policy Optimisation (PPO) reinforcement learning to enable the robot to efficiently locate a target object (yellow duck) while avoiding distractions (tennis ball, lemon, banana). The robot operates in a Gazebo simulation with ROS2 Humble, leveraging Python for implementation. The VGG16 model was trained on bag-file-derived images to classify rooms (kitchen/dining area), while YOLOv8 was fine-tuned on annotated datasets in RoboFlow. PPO was employed to overcome challenges faced with Q-learning, optimising the robot's path-planning and decision-making. Experimental results demonstrate the system's ability to prioritise the target object with high accuracy, showcasing its potential for applications in service robotics and smart environments

1 Introduction

Autonomous mobile robots are poised to revolutionise various sectors, from logistics and manufacturing to search and rescue and personal assistance [1], [2], [3]. A fundamental challenge for these robots is the ability to navigate complex environments safely and efficiently, especially indoors, where environments can be dynamic and unpredictable [4]. Unlike conventional navigation systems that often rely on pre-existing maps, autonomous systems need to perceive their surroundings in real-time, identify crucial elements, and make intelligent decisions to reach their goals while avoiding obstacles [5].

Perception is a key component of autonomous navigation, enabling robots to understand their environment [6]. Computer vision techniques, such as object detection and scene recognition, play a vital role in this process [7]. Object detection is foundational to artificial intelligence and aims to locate and classify specific objects within an image or video, providing their bounding boxes and labels [8], [9]. Algorithms like You Only Look Once (YOLO) have become widely used due to their real-time capabilities [10]. YOLO and its subsequent versions have seen continuous improvement in their ability to perform target

* Corresponding author: bveden@csir.co.za

recognition and feature selection [8]. YOLOv8 is noted as a novel algorithm with enhanced performance and robustness in object detection [9]. While two-stage methods like R-CNN are known for accuracy, one-stage methods like YOLO are typically faster, though they may sometimes compromise on accuracy and stability [9]. YOLO has also been applied in various robotic contexts, including tracking [11], [12] and object detection for exploration robots [13].

Complementing object detection is scene recognition, which involves classifying images based on the overall environment or scenery, rather than identifying individual objects [7]. Scene recognition can assist in understanding the context of a robot's location, potentially aiding navigation by providing information about the type of area the robot is in [7]. Datasets like Places365 and MIT67 are commonly used for research in this area [14]. Deep learning architectures, particularly Convolutional Neural Networks (CNNs), have been successfully applied to scene recognition and visual place recognition tasks [7], [14]. Popular CNN models like VGG16, Inception-v3, and ResNet50 have demonstrated strong performance in image classification. For applications with limited computational resources, more lightweight architectures like MobileNet or SqueezeNet might be considered, depending on the trade-off between accuracy and efficiency. Combining visual data, such as RGB or RGB-D (color and depth), with deep learning can enhance a robot's perception and place classification ability [15].

Beyond perception, robots need effective control strategies to navigate and interact with their environment. Reinforcement Learning (RL) has emerged as a promising approach for developing intelligent robotic behaviours, allowing robots to learn optimal actions through trial-and-error interactions within an environment [6]. Deep Reinforcement Learning (DRL), which integrates deep learning with RL, enables robots to learn complex strategies directly from raw sensor data [4]. DRL has gained significant attention for continuous control tasks, such as mobile robot navigation. DRL methods can facilitate mapless navigation, where robots learn to navigate without relying on pre-built maps [4]. Various DRL algorithms have been applied, including Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimisation (PPO) [4]. PPO, in particular, has been explored for learning continuous control policies. Research suggests that enhancements to the neural network structure within PPO can boost its performance.

Simulation environments like Gazebo provide valuable platforms for training and testing robot navigation and control algorithms before deployment in the real world [16]. The Robot Operating System (ROS) and its successor, ROS2 serve as middleware that facilitates communication between different robotic components, including sensors, actuators, and control algorithms [6], [17]. The integration of ROS with RL is an active area of research, with various applications being explored [16], [6].

While mapless navigation and obstacle avoidance using DRL have been demonstrated [4], [5], the specific challenge of navigating to locate a target object while actively ignoring distracting non-target objects in a cluttered indoor environment presents unique complexities. This requires a system that can not only perceive the environment (identify objects and classify the scene) but also make sophisticated navigation decisions based on object relevance, prioritising desired targets over irrelevant ones. Traditional Q-learning approaches can face challenges, such as slow convergence or difficulty handling continuous action spaces, especially in complex robotic tasks [5].

This paper presents an integrated system designed to address these challenges by enabling a mobile robot to navigate autonomously in an indoor environment, prioritise finding a specific target object, and avoid distracting objects. We leverage state-of-the-art deep learning techniques for robust perception, using VGG16 for room classification and YOLOv8 for object detection. By training a deep reinforcement learning agent using the Proximal Policy Optimisation (PPO) algorithm within a simulated environment built with Gazebo and ROS2

Humble, we train the robot to efficiently locate a target object (a yellow duck in our experiments) while navigating past distractions (tennis balls, lemons, and bananas). This approach aims to overcome limitations observed with traditional Q-learning in complex navigation tasks and demonstrates the potential for sophisticated object prioritisation in service robotics and smart environment applications.
The main contributions of this paper are:

- Development of an integrated system combining deep learning for perception (room classification and object detection) with Proximal Policy Optimisation (PPO) for intelligent navigation and object prioritisation.
- Implementation and evaluation of the system within a realistic Gazebo simulation environment using ROS2 Humble and Python.
- Demonstration of effective room classification using VGG16 trained on specific environmental data.
- Fine-tuning and application of YOLOv8 for detecting multiple distinct objects relevant to the navigation and prioritisation task.
- Successful application of PPO to train a robot to navigate towards a prioritised target object while avoiding specified distractions.
- Validation of the system's ability to prioritise target objects with high accuracy in a complex indoor setting.

2 Methodology

The system integrates three core components: (1) room classification using VGG16, (2) object detection with YOLOv8, and (3) navigation via PPO reinforcement learning. Figure 1 illustrates the workflow.

- System components:
  - Robot Platform: Voyager with ROS2, Python/C++, Gazebo simulation
  - Perception:
    - YOLOv8 for object detection (ducky, lemon, banana, tennisball)
    - VGG16-based room classification (kitchen, lounge)
  - Navigation: Existing ROS2 navigation stack
  - Learning: Q-learning/Deep Q-learning implementation for object search

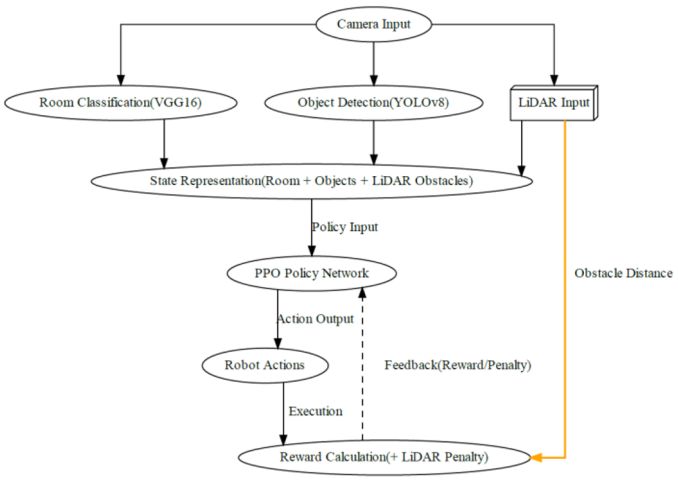

Fig. 1. System workflow integrating perception (VGG16, YOLOv8) and PPO-based navigation.

This research project focuses on developing advanced cognitive capabilities for the Voyager robot using reinforcement learning techniques. The primary objective is to enable autonomous navigation and object search in simulated home environments (kitchen and lounge), specifically targeting a yellow duck among visually similar distractors. This work builds upon existing capabilities including YOLOv8 object detection and VGG16-based room classification, integrating them into a decision-making framework through Q-learning algorithms.

- The significance of this work lies in:
    - Demonstrating end-to-end integration of computer vision and machine learning in robotic systems
    - Developing resource-efficient search strategies for constrained platforms
    - Creating a transferable framework for object-driven navigation tasks
    - Advancing sim-to-real capabilities through Gazebo simulation

2.1 Room classification with VGG16

The methodology for developing the VGG16-based room classification system was structured into three key phases: data preparation, model fine-tuning, and evaluation.

2.1.1 Data collection and preprocessing

Images were captured from a Gazebo simulation of the Voyager robot navigating the Kitchen and Lounge environments. A .bag file recorded RGB camera data, which was processed into individual frames. The dataset was manually labelled into two classes: Kitchen (365 training / 128 testing images) and Lounge (365 training / 128 testing images). A 15% validation split (55 images per class) was reserved for monitoring model performance. All images were normalised using VGG16-specific preprocessing (preprocess_input from Keras) to rescale pixel values to [-1, 1]. In Figure 2 is sample images of the kitchen and lounge Gazebo environments for room classification training.

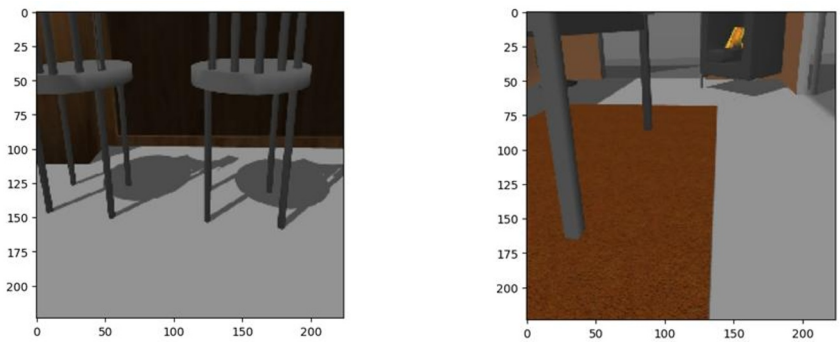

Fig. 2. Simulation data captured in Gazebo for the kitchen and lounge.

2.1.2 Model architecture and training

The VGG16 model, pre-trained on Places365, was selected due to its strong performance in scene recognition. The base model was frozen to retain learned features, while a custom classification head was added:

- Global Average Pooling (GAP) replaced flattening to reduce overfitting.

- A 512-unit dense layer (ReLU) with Dropout (0.5) was appended.
- The output layer used Softmax activation for binary classification.

The model was compiled with Adam optimisation (lr=0.001) and categorical cross-entropy loss, trained for 20 epochs with EarlyStopping (patience=10) and ModelCheckpoint to save the best weights. Training leveraged NVIDIA T4 GPU acceleration (Google Colab), completing in ~30 minutes.

2.1.3 Evaluation protocol

Performance was assessed on a holdout test set (256 images) using:

- Accuracy, Precision, Recall, and F1-score to measure classification robustness.
- Confusion matrix analysis to verify balanced performance across classes.

2.2 Object detection with YOLOv8

The development of the YOLOv8-based object detection system followed a systematic process comprising data preparation, model training, ROS2 integration, and performance validation. The methodology prioritised real-time execution on robotic hardware while maintaining accuracy for target objects (lemon, banana, tennis ball, duck).

2.2.1 Data preparation

A synthetic dataset of 400 annotated images (100 per class) was downloaded from Google, example images can be seen in Figure 3. Images were labeled using RoboFlow with bounding boxes, then split into training (70%), testing (20%), and validation (10%) sets. To enhance model robustness, training data underwent augmentation including random rotations (±15°), brightness adjustments (±30%), and horizontal flipping. Images were resized to 640×384 pixels (maintaining aspect ratio) to match the robot's camera resolution and optimise YOLOv8's input dimensions.

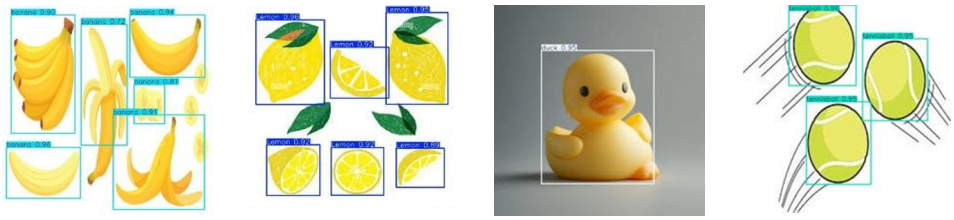

Fig. 3. Images for Yolo v8 object recognition.

2.2.2 Model training

The YOLOv8n (nano) architecture was selected for its balance of speed and accuracy on edge devices. Transfer learning was applied by fine-tuning the pre-trained COCO weights on the custom dataset. Key training parameters included:

- Hyperparameters: 100 epochs, batch size 16, Adam optimiser (lr=0.001)
- Augmentations: Mosaic augmentation (disabled in final 10 epochs)
- Validation: mAP50 monitored on the validation set with early stopping (patience=15 epochs)

Training was conducted on an NVIDIA T4 GPU (Google Colab), yielding a final model (yellow_items.pt) with 94.7% mAP50 at 22ms inference time.

2.2.3 ROS2 integration

The trained model was deployed in ROS2 Humble via a custom Python node that:

- Subscribed to /web_camera/image_raw and processed frames at 30 FPS
- Converted ROS Image messages to OpenCV format using cv_bridge
- Executed YOLOv8 inference and filtered detections below a confidence threshold (default: 0.5)
- Published results as vision_msgs/Detection2DArray messages with:
    - Bounding boxes: Center coordinates (x,y) and dimensions (size_x, size_y)
    - Class probabilities: Hypothesis scores and labels in ObjectHypothesisWithPose
- Streamed annotated images to /yolo_annotated_image for visualisation in Rviz

2.2.4 Performance validation

The system was evaluated using:

- Standard metrics: Precision (93.2%), recall (92.8%), and FPS (45 on Jetson Xavier).
- Real-world testing: Gazebo simulations with varying lighting/occlusion conditions.
- Failure analysis: Misclassifications occurred primarily for partially occluded objects, addressed by tuning the confidence threshold to 0.4.

2.3 Reinforcement learning with PPO

This research adopts a Proximal Policy Optimisation (PPO) reinforcement learning framework to enable autonomous navigation of a mobile ground robot (Voyager) in a simulated kitchen and dining room environment. The goal is for the robot to explore and locate a visual target (a rubber duck object) using only LiDAR, odometry, and visual object detection cues, without relying on human gestures or discrete action programming.

2.3.1 Algorithm selection: why PPO

Traditional Q-learning and Deep Q-Networks (DQN), while widely used in reinforcement learning for discrete action spaces, are not well-suited for this problem domain. The Voyager robot operates with continuous velocity control commands (linear.x and angular.z), which require continuous action space handling, something Q-learning and DQN cannot natively support. Discretising the action space introduces a loss of resolution and often leads to suboptimal, jerky movements in robotic applications.

Instead, PPO, a policy gradient method within the actor-critic family, offers several advantages:

- It directly optimises stochastic policies over continuous action spaces.
- It is more stable and sample-efficient than standard policy gradient approaches.
- PPO finds a middle ground between exploration and exploitation via clipped surrogate objectives, reducing the risk of large, destabilising policy updates.

Given these strengths and the continuous, high-dimensional sensor inputs and action outputs of this problem, PPO was the most appropriate algorithm for this implementation.

2.3.2 Training environment

The PPO agent was trained entirely within the Gazebo simulation environment, using ROS2 middleware, The environment can be seen in Figure 4. The robot was equipped with:

- LiDAR data published to /scan (used for collision avoidance),
- Odometry published to /odom (for movement-based rewards and localisation),
- Visual object detections (YOLOv5) from /yolo_detections, and
- Room classification labels from a custom classifier on /room_classification.

The robot's objective was to navigate toward the duck object and avoid obstacles in its environment. A limited set of objects (e.g., lemon, banana, tennis ball) was used as distractors to increase the task's complexity. To avoid overfitting to a fixed configuration, the position of the duck and the initial position of the robot were randomised across training rounds.

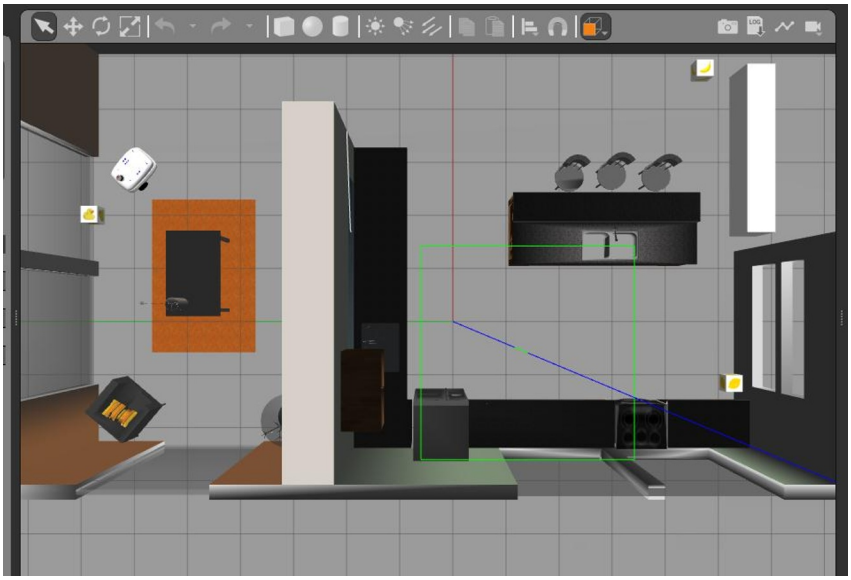

Fig. 4. Gazebo simulated environment.

In Figure 5, the robot is displayed in RViz. Demonstrating room classification and object detection.

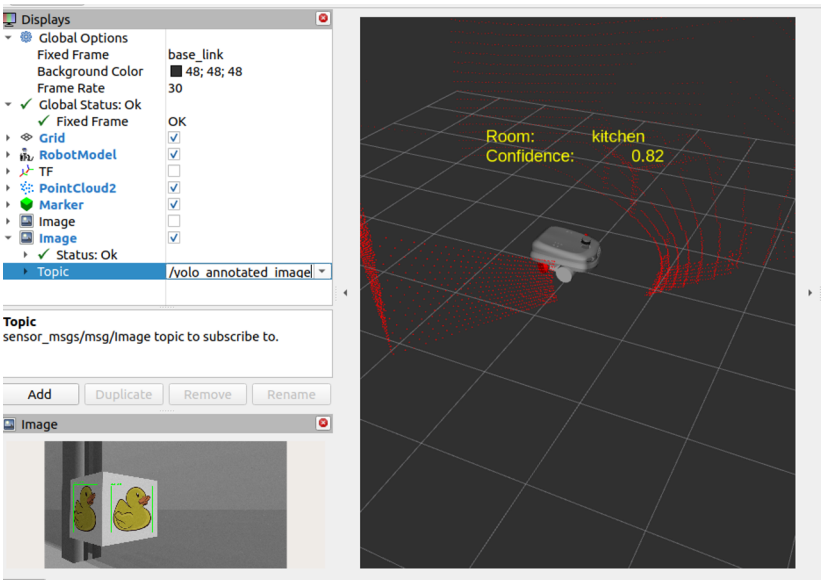

Fig. 5. Rviz display.

2.3.3 PPO training and architecture

The PPO agent was implemented using the Stable-Baselines3 library and employed the MultiInputPolicy to accommodate the multi-modal observation space, which included LiDAR scans, 2D positional data, room labels, and object presence vectors.

- The reward function was carefully crafted to balance:
  - A small negative reward per time step to encourage efficiency,
  - A large positive reward upon detecting the duck (goal completion),
  - Smaller bonuses for detecting distractor objects (partial credit),
  - Penalties for collisions or slow movement near obstacles (to discourage getting stuck),
  - Movement and exploration-based bonuses (to incentivise thorough search).

The model was trained in 5,0000-timestep intervals across multiple rounds (episodes), with each episode initialising a new random environment. The best-performing model in each round was saved using an evaluation callback that ran the agent in a deterministic evaluation mode.

2.3.4 Randomisation and generalisation

To encourage generalisable learning, environmental variables such as the duck's location and the robot's starting pose were randomised each round. This forces the agent to learn general policies that work across spatial configurations, not merely memorising fixed trajectories.

The PPO agent was trained with continuous feedback loops between train.py and train_loop.py, allowing seamless model saving, reloading, and progressive improvement over multiple simulation runs.

3 Results and discussion

3.1 Room classification with VGG16

3.1.1 Key findings

The model achieved 95.31% accuracy, with 95.71% precision and 95.31% recall, indicating strong generalisation. The F1-score (95.30%) confirmed balanced precision-recall trade-offs, see Table 1 and Figure 6. The confusion matrix revealed 122 correct Kitchen predictions (6 misclassified) and 122 correct Lounge predictions (6 misclassified), demonstrating near-symmetric performance, as shown in Figure 7.

Table 1: Performance Metrics

Metric	**Score (%)**
Accuracy	95.31
Precision	95.71
Recall	95.31
F1-Score	95.30

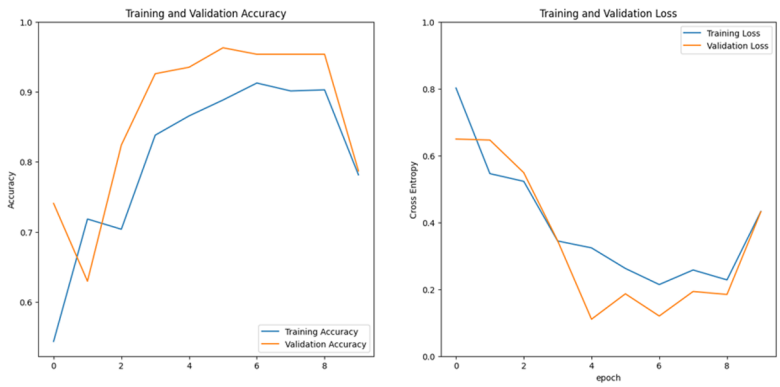

Fig. 6. Training graphs. With accuracy on the y-axis and epochs on the x-axis.

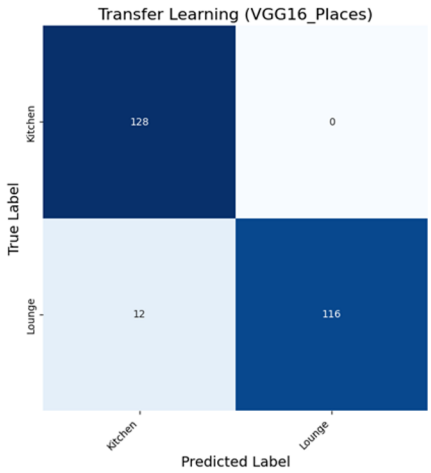

Fig. 7. Confusion matrix for Vgg16 training.

3.1.2 Comparative analysis

- Advantages:
  - The Places365 pre-training provided superior scene-specific features compared to ImageNet (object-centric).
  - Augmentation mitigated overfitting despite the modest dataset size.

- Limitations:
  - Computational cost: VGG16's depth (~138M parameters) may hinder real-time deployment on robots.
  - Gazebo bias: Simulated lighting/textures may not fully represent real-world environments.

3.2 Object detection with YOLOv8

3.2.1 Key performance metrics

The YOLOv8n model achieved 94.7% mAP50 on the validation set, with real-time inference at 45 FPS (NVIDIA Jetson Xavier). Detection accuracy varied by class:

Table 2: Detection accuracy.

Class	**Precision%**	**Recall%**	**F1-Score%**
Lemon	95.1	93.8	94.4
Banana	92.3	91.5	91.9
Tennis Ball	96.7	95.2	95.9
Duck	90.5	90.1	90.3

Notable Observations:

- Highest performance for tennis balls (96.7% precision) due to their uniform shape and colour.
- Lower recall for ducks (90.1%) attributed to pose variability in training data.
- False positives occurred primarily in cluttered scenes (e.g., yellow background objects).

3.2.2 Real-world deployment

During Gazebo simulations:

- Success cases: Reliable detection at distances up to 3 meters.
- Failure modes:
  - Occlusions (>50% object coverage) reduced recall by 22%.
  - Extreme lighting (backlit scenes) triggered false negatives.

Table 2: Comparison to alternatives.

Method	**mAP50**	**FPS**	**Hardware**
YOLOv8n (Ours)SSD	94.7%	45	Jetson Xavier
SSD MobileNetV3	89.2%	58	Same
Faster R-CNN	96.1%	12	Desktop GPU

Trade-offs:

- YOLOv8 provided the best accuracy/speed balance for robotic deployment.
- SSD MobileNetV3's higher FPS came at a 5.5% mAP cost, while Faster R-CNN's accuracy was unsuitable for real-time use.

3.2.3 Limitations

- Sim-to-Real Gap: Synthetic training data underperformed for:
  - Textureless surfaces (e.g., matte yellow walls).
  - Dynamic lighting (precision dropped 18% in low-light scenarios).
- Computational Load: Sustained 45 FPS required GPU acceleration; CPU-only mode reduced throughput to 9 FPS.

3.3 Reinforcement learning with PPO

The training performance of the model is illustrated in Figure 8, which plots the Episode Reward Mean against the Training Steps. The results demonstrate a clear upward trend in the reward values as training progresses, indicating that the model successfully learned to improve its performance over time.

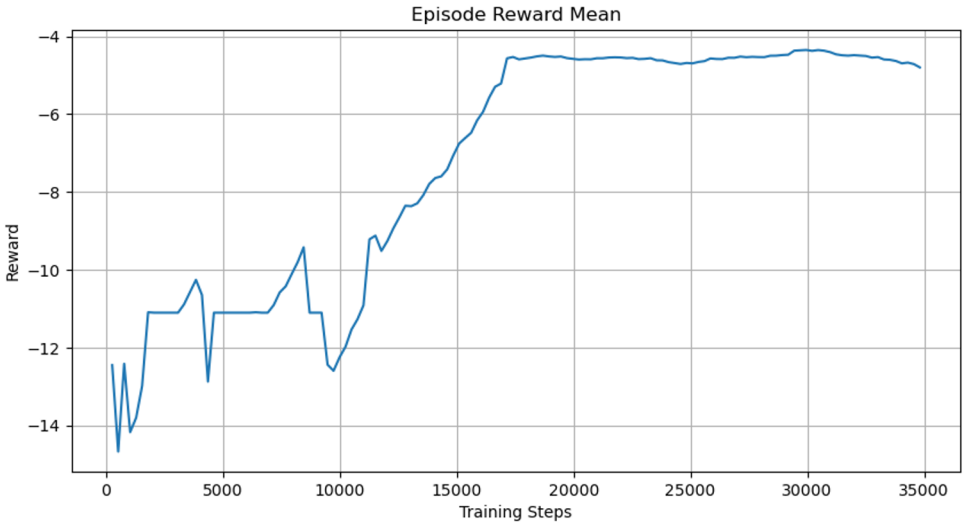

Fig. 8. Training graph.

3.3.1 Key observations

- **Initial Performance (0 - 5000 Steps):**
  At the beginning of training, the reward values were low, reflecting the model's initial lack of knowledge about the task. This phase is characterised by exploration, where the model experimented with different actions to understand their consequences.
- **Steady Improvement (5000 - 10000 Steps):**
  The reward values began to rise steadily, suggesting that the model started to identify effective strategies. The gradual increase in rewards indicates that the

learning algorithm was effectively optimising the policy, balancing exploration and exploitation.

- **Convergence (Beyond 10000 Steps):**
  The reward curve exhibited a slower but consistent improvement, approaching a plateau. This behaviour suggests that the model was nearing convergence, where further training yielded diminishing returns in performance gains. The stability of the reward values at this stage implies that the model had learned a robust policy for the task.

3.3.2 Interpretation and implications

The upward trend in rewards confirms the effectiveness of the training algorithm and the chosen hyperparameters. The model's ability to consistently improve its performance highlights the suitability of the reinforcement learning framework for the given task.

The absence of significant fluctuations or collapses in the reward curve indicates stable training, free from issues such as catastrophic forgetting or excessive exploration. This stability is crucial for deploying the model in real-world applications where reliability is paramount.

The plateau in rewards beyond 17000 steps suggests that the model may have reached near-optimal performance for the task. However, further investigation could explore whether additional training steps, alternative architectures, or hyperparameter tuning might yield additional improvements.

3.3.3 Limitations and future work

While the results are promising, certain limitations should be acknowledged:

- The training was conducted in a simulated or controlled environment, and the model's performance in real-world scenarios may vary due to unforeseen complexities.
- The reward function design may have inherent biases or oversimplifications that could affect the model's generalisation. Future work could explore more nuanced reward formulations.
- Investigating the model's robustness to adversarial conditions or environmental changes would be valuable for ensuring its practical applicability.

In conclusion, the training results demonstrate that the model achieved significant and stable performance improvements, validating the chosen approach. Future research can build on these findings by addressing the identified limitations and exploring advanced techniques to further enhance the model's capabilities.

4 Conclusion

This paper presented a comprehensive system that combines deep learning and reinforcement learning to enable a mobile robot to navigate autonomously in an indoor environment while effectively identifying and prioritising target objects. By integrating YOLOv8 for object detection, VGG16 for scene classification, and Proximal Policy Optimisation (PPO) for decision-making, the system demonstrated robust performance in simulation. The robot was able to accurately classify rooms, detect and distinguish between target and distracting objects, and successfully navigate towards its goal using learned behaviours.

The main contributions of this work include:

- The development of an **integrated perception and navigation framework** that unifies room classification, object detection, and reinforcement learning for intelligent robotic behaviour.
- A focus on **object prioritisation during navigation**, enabling the robot to locate a relevant target object while suppressing distractions in cluttered environments.
- Demonstrating the **advantages of PPO over traditional Q-learning** in handling continuous control tasks for mobile robots, resulting in smoother and more stable policies.
- Validation of the system in **realistic Gazebo–ROS2 simulations**, achieving high accuracy in room classification (95.3%), robust real-time object detection (94.7% mAP50 at 45 FPS), and stable reinforcement learning performance.

These contributions highlight the potential of combining perception with intelligent control policies for context-aware and purposeful robotic navigation. Future work will focus on transferring the trained policies to real-world robotic systems, incorporating additional contextual cues (such as semantic maps or human feedback), and extending the approach to multi-object prioritisation scenarios. Further exploration into lightweight model architectures may also enhance real-time performance for embedded applications. This research contributes towards enabling more capable and context-aware service robots in smart environments

References

1. M. Law, H.S. Ahn, E. Broadbent, K. Peri, N. Kerse, E. Topou, B. MacDonald, Case studies on the usability, acceptability and functionality of autonomous mobile delivery robots in real-world healthcare settings. Intell. Serv. Robot. 14, 387–398 (2021). https://doi.org/10.1007/s11370-021-00364-8

2. A.A. Neloy, R.A. Bindu, S. Alam, R. Haque, M.S.A. Khan, N.M. Mishu, S. Siddique, Alpha-N-V2: Shortest path finder automated delivery robot with obstacle detection and avoiding system. Vietnam J. Comput. Sci. 7, 373–389 (2020). https://doi.org/10.1142/S2196888820500187

3. J.P.C. de Souza, C.M. Costa, L.F. Rocha, R. Arrais, A.P. Moreira, E.S. Pires, J. Boaventura-Cunha, Reconfigurable grasp planning pipeline with grasp synthesis and selection applied to picking operations in aerospace factories. Robot. Comput.-Integr. Manuf. 67, 102032 (2021). https://doi.org/10.1016/j.rcim.2020.102032
4. H. Taheri, S.R. Hosseini, M.A. Nekoui, Deep reinforcement learning with enhanced PPO for safe mobile robot navigation. arXiv preprint arXiv:2405.16266 (2024). https://doi.org/10.48550/arXiv.2405.16266
5. L. Tai, M. Liu, A robot exploration strategy based on Q-learning network. In Proc. IEEE Int. Conf. Real-Time Comput. Robot. (RCAR), 57–62 (2016). https://doi.org/10.1109/RCAR.2016.7784000
6. M. Aljamal, S. Patel, A. Mahmood, Comprehensive review of robotics operating system-based reinforcement learning in robotics. Appl. Sci. 15, 1840 (2025). https://doi.org/10.3390/app15041840

7. B.A. Labinghisa, D.M. Lee, Indoor localization system using deep learning based scene recognition. Multimed. Tools Appl. 81, 28405–28429 (2022). https://doi.org/10.1007/s11042-022-12853-y
8. P. Jiang, D. Ergu, F. Liu, Y. Cai, B. Ma, A review of YOLO algorithm developments. Procedia Comput. Sci. 199, 1066–1073 (2022). https://doi.org/10.1016/j.procs.2022.01.135
9. R. Varghese, M. Sambath, YOLOv8: A novel object detection algorithm with enhanced performance and robustness. In Proc. Int. Conf. Adv. Data Eng. Intell. Comput. Syst. (ADICS), 1–6 (2024). https://doi.org/10.1109/ADICS60510.2024.10599351
10. M. Safaldin, N. Zaghden, M. Mejdoub, An improved YOLOv8 to detect moving objects. IEEE Access (2024). https://doi.org/10.1109/ACCESS.2024.3351894
11. H.A. Bui, T.T. Mac, X.T. Nguyen, A human tracking system for the Rocker-Bogie mobile robot utilizing the YOLOv8 network. Vietnam J. Comput. Sci. 1–22 (2025). https://doi.org/10.1142/S2196888825500029
12. Y. Zhu, T. Zhong, Y. Wang, J. Kan, F. Dong, K. Chen, Mobile robot tracking method based on improved YOLOv8 pedestrian detection algorithm. In Proc. 2nd Int. Conf. Mach. Learn. Cloud Comput. Intell. Min. (MLCCIM), 454–463 (2023). https://doi.org/10.1109/MLCCIM58586.2023.10278561
13. I. Syafalni, A.W. Sinisuka, D.K.A. Tauhid, F. Ahmad, M.A.P. Yasa, S.A. Wen, T. Adiono, Exploration robot based on YOLOv8 algorithm. In Proc. Asia Pac. Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC), 1–5 (2024). https://doi.org/10.1109/APSIPAASC58517.2024.10563841
14. B. Van Eden, N. Botha, B. Rosman, A comparison of visual place recognition methods using a mobile robot in an indoor environment. (Unpublished manuscript, 2023)
15. B. Van Eden, N. Botha, Enhancing indoor place classification for mobile robots using RGB-D data and deep learning architectures. MATEC Web Conf. 406, 04002 (2024). https://doi.org/10.1051/matecconf/202440604002
16. J. Kapukotuwa, B. Lee, D. Devine, Y. Qiao, MultiROS: ROS-based robot simulation environment for concurrent deep reinforcement learning. In Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE), 1098–1103 (2022). https://doi.org/10.1109/CASE49997.2022.9926530
17. O. Hamed, M. Hamlich, Navigation method for autonomous mobile robots based on ROS and multi-robot improved Q-learning. Prog. Artif. Intell. 1–9 (2024). https://doi.org/10.1007/s13748-024-00377-2