Elevating UAV Multi Object Tracking Capabilities: Synergistic Integration of YOLOv8, Nano Instance Segmentation, and Dueling Double Deep Q Network

R Kiruthiga[a,*], **B Nithya**[a], **S Martin Prabhu**[b]
[a]Department of Computer Science and Engineering, National Institute of Technology, Tiruchirapalli, India
[b]Department of Data Science, St.Joseph's College, Tiruchirapalli, India

Abstract. Unmanned Aerial Vehicles (UAVs) play a pivotal role in navigating intricate landscapes, reaching remote or hazardous areas, and capturing high-resolution imagery. This paper introduces an advanced methodology for object detection, particularly within the framework of computer vision applied to UAVs. Traditional approaches, employing deep learning models like RCNN, Fast RCNN, and YOLO, have inherent limitations such as struggling with predicting occluded, blurred, or clustered objects, and the inability to identify and track multiple objects simultaneously. To address these limitations, YOLOv8x (You Only Look Once), Nano instance segmentation (NIS) and Dueling Double Deep Q Network (DDDQN) are integrated by the proposed framework. YOLOv8x demonstrates superior performance, achieving an impressive Average Precision (AP) of 53.9% on the challenging test set of the MSCOCO dataset, surpassing its predecessors. The DDDQN algorithm enhances tracking capabilities by efficiently estimating state values and state-dependent action advantages independently. The fusion of YOLOv8x with the DDDQN architecture enables adept management of obstacles, variations in object size, and unexpected movements. The proposed framework is simulated with UAVDT and VisDrone datasets and compared with roughly nine existing frameworks that have been recently proposed in the literature. From this, it is observed that the proposed framework enhances object tracking in densely populated environments, offering a promising solution for real-world applications that demand precise and resilient object detection.

Keywords: UAVs, Object Tracking, Detection, YOLOV8, Nano Instance Segmentation, DDDQN.

*R Kiruthiga, kiruthigarajendran90@gmail.com

1 Introduction

In recent years, the integration of Unmanned Aerial Vehicles (UAVs) into various industries has witnessed unprecedented growth, driven by advancements in sensor technologies, flight capabilities, and data processing algorithms. The ability of UAVs to navigate complex environments, access remote or hazardous areas, and capture high-resolution imagery makes them an ideal platform for tasks requiring a bird's-eye perspective. Within the domain of computer vision, object tracking emerges as a pivotal application, involving the identification and continual monitoring of objects as they traverse diverse camera perspectives. This technology possesses the capability to recognize and trail multiple objects within an image. For instance, in a football broadcast scenario, this technology could be employed to trace the trajectory of the ball in a photograph.Object detection and tracking finds its application in identifying and tracking objects in disaster-stricken areas for efficient resource allocation.,[1] Monitoring bird populations for ecological studies or to prevent bird strikes in aviation,[2] monitoring crop health, detecting pests, and optimizing farming practices,[3] identifying and tracking unauthorized drones to ensure airspace security,[4]Enhancing traffic management through the detection and tracking of vehicles[5] and it can also be employed in various civil engineering applications, such as monitoring construction sites and infrastructure.[6] Though it has many applications, it comes with the following challenges: *(i) Scale and Orientation Variability:* Objects may vary in size and orientation, making it challenging to develop a one-size-fits-all tracking algorithm. *(ii) Occlusions:* Objects may be partially or fully occluded, leading to tracking failures. *(iii) Real-time Processing:* Many applications require real-time tracking, necessitating efficient algorithms for rapid processing. *(iv) Adaptability to Environments:* Tracking algorithms should be adaptable to different environmental conditions, such as varying lighting and weather. *(v) Data Annotation:* Supervised learning for object tracking requires annotated datasets, which can be labor-intensive and expensive to create.

The standard solutions for the above mentioned challenges are Kalman Filters, Mean-Shift Algorithm, Correlation Filters, etc,. However, these traditional methods may struggle with handling complex scenes, occlusions, and object appearance changes. Manual feature engineering is also required, making the algorithms less adaptive to diverse datasets. To address these issues, Deep Learning(DL) based solutions are proposed in the literature. DL models automatically learn hierarchical representations from data, eliminating the need for manual feature engineering. It excels at capturing intricate patterns and variations in object appearance. DL-based trackers often outperform traditional methods in terms of accuracy. DL models can also adapt to diverse scenes and handle variations in object appearance. Pre-trained models can be fine-tuned for specific tracking tasks, reducing the need for large labeled datasets.

Among the various DL techniques, some of basic techniques are listed as follows: *(i) Convolutional Neural Networks (CNNs)*, which are especially effective in image-related tasks and adept at feature extraction, *(ii) Recurrent Neural Networks (RNNs)* which are suitable for tracking sequential data and it can capture temporal dependencies, *(iii) Long Short-Term Memory (LSTM)* which is a type of RNN designed to overcome the vanishing gradient problem, useful for tracking over long sequences, *(iv) Models like Faster R-CNN, YOLO (You Only Look Once), and SSD (Single Shot Multibox Detector)* are commonly used for object detection and tracking. Traditional object detection and tracking algorithms using deep learning approaches like RCNN, Fast RCNN, FPN, YOLO have certain limitations associated with it. They are unable to predict the occluded, blurred or clustered objects and also its does not identify and track multiple objects in the image.[7]

To overcome all these drawbacks, the framework namely Nano Instance Segmentation with DDDQN (NISD) is proposed in this paper which integrates three different techniques namely YOLOV8x for detecting bounding boxes, Nano instance segmentation for predicting the masks in the bounding boxes and finally Dueling Double DQN(DDDQN) algorithm for tracking multiple objects. YOLOv8 shows better performance when compared to all other previous YOLO models. Assessed on the challenging test set of the MSCOCO dataset, YOLOv8x accomplished an extraordinary AP of 53.9%, surpassing the 50.7% attained by YOLOv5 at the same input resolution of 640 pixels. Astonishingly, YOLOv8x exhibited remarkable speed, clocking in at an impressive 280 frames per second on NVIDIA's powerful A100 GPU in conjunction with TensorRT.[8] In the Dueling Network Architecture, the value and advantage functions are estimated separately. The idea behind this separation is to improve the efficiency of Q-value estimation in deep reinforcement learning by allowing the network to focus on learning state values and state-dependent action advantages independently. In this architecture, there are two separate streams in the neural network: one for estimating the state value and another for estimating the state-dependent action advantages. By doing this, the network can learn which states are valuable (the value stream) and which actions are advantageous within those states (the advantage stream).[9]This separation allows the network to generalize better and learn faster, ultimately improving the overall performance of deep reinforcement learning algorithms.The fusion of YOLOv8x with DDDQN architecture enables skillful management of obstructions, alterations in size, and unexpected movement, resulting in improved tracking capabilities.

The subsequent sections are organized as follows: Section 2 provides an extensive review of prior research in the domains of object detection, tracking, and reinforcement learning-based approaches. Section 3 introduces our proposed approach, detailing the integration of YOLOv8, Segmentation map and Dueling Double DQN (DDDQN) framework. Section 4 explains the simulation parameters, comparision metrics and results. Finally, Section 5 concludes the paper and highlights the potential avenues for future research.

2 Related Works

The comprehensive object detection and tracking methodology presented in this section significantly enhances insight about the recent object tracking techniques in the literature.

An algorithmic framework in[10] capitalizes on a cutting-edge Deep Learning neural network (YOLOv2) as its central processing block, but it also harnesses navigation data from both the tracker and target UAVs to prophesy the target's position on the image plane. This approach streamlines computational time and curtails the occurrence of false alarms that may arise when scouring the entire image plane for the target.The architecture faces difficulties in correctly identifying targets that have low contrast with the cluttered background. The authors[11] unveil an avant-garde multiple object tracking system that employs an enhanced object detection framework fusing YOLOv3 which grapples with minuscule objects that congregate together and RetinaNet for detecting small size objects. To foreshadow object movement, the authors opt for a simplistic filter, and for data association, they employ a profound association matrix generated by a CNN model pre-conditioned on the MARS dataset. However, its operation requires lightning-fast computer systems and may confront challenges when detecting minuscule objects or dealing with occlusions. An innovative approach named the Improved Dueling Deep Double-Q Network Based on Prioritized Experience Replay (IPD3QN) is introduced in[12] to address the challenges of slow and erratic convergence typically observed in traditional Deep Q Network (DQN) algorithms when applied to the autonomous path planning of Unmanned Surface Vehicles (USVs).The experience replay unit embraces the prioritized experience replay approach to harvest experience samples, amplifying the employment of real samples and hastening the velocity of neural network training. Subsequently, a dueling network structure is introduced to optimize the neural network. Lastly, the soft update technique is employed to enhance the algorithm's stability, while the dynamic ϵ greedy method is enlisted to uncover the paramount strategy. Areas of improvements include convergence speed and stability in path planning under complex continuous action settings.

In,[13] an all-encompassing autonomous object tracking UAV system with YOLOv4Tiny is tailored for surveillance applications. By leveraging deep learning-based semantic object detection, 3D object pose estimation, and the Kalman filter, along with UAV path planning, the system accomplishes the capacity to execute surveillance tasks without human intervention. The affirmative outcomes of the flight experiments substantiate the practical viability of the proposed system for real-world surveillance applications. The objective of the work in[14] is to refine vehicle identification, tracking, and geolocation in UAS-acquired images by harnessing recognitions, location, and tracking features to magnify decision-making. The study unveils an ingenious, self-progressive methodology for these tasks, employing reinforcement learning based on a Fast RCNN algorithm, thus embodying the potential for precise vehicle identification. The pioneering approach exhibits triumphant performance in arduous scenarios, such as densely congested parking garages, bustling intersections, and teeming streets.

The technique[15] employs a synergistic recurrent neural network-based Deep Q-Network (DQN) tracker to forecast and trail objects while acquiring knowledge about the environment. The model is honed using virtual image-based video sequences. The authors envisage enhancing the model's performance through the implementation of fine-tuning methodologies and conducting extensive simulation experiments under diverse weather conditions. Furthermore, they intend to subject the model to other open-source video sequences to compare its efficacy with conventional RL-based DQN trackers.

The paper[16] focuses on the development of an object detection algorithm for drones operating in large-scale maritime scenarios. The proposed novel object detection algorithm uses the YOLOv5s framework and it leverages the Transformer architecture to enhance feature extraction, especially for detecting small or occluded objects.The evaluation of GGT-YOLO is grounded in specific datasets, encompassing the MariDrone and RSOD datasets. However, the algorithm's performance on alternative datasets or within diverse environments may manifest dissimilarly. The algorithm's performance under arduous conditions such as feeble lighting, adverse weather, or cluttered backdrops may not have been exhaustively evaluated.

The Lightweight Deep Vision Reinforcement Learning (LDVRL) framework[17] incorporates a medley of techniques, including stacks of frames, segmentation maps, and depth images, to curtail computational expenditures. The experiments were conducted with two Profound Reinforcement Learning (DRL) approaches, a non-sparse Deep Q-Network (DQN) and a Deep Deterministic Policy Gradient (DDPG)and it was discovered that the DQN outperforms the DDPG in this particular undertaking. The selection of the DQN was predicated on its reduced model dimensions, which diminish computational resources, and its capability to function in an unbroken state space. The proposed framework is suitable only for multirotar UAVs.

The goal of UAVMOT[18] is to efficiently and accurately track multiple objects in the dynamic and challenging environment of UAV videos. The proposed network addresses several key challenges specific to tracking in UAV videos, such as complex UAV motions, limited hardware on mobile UAV equipment, and the need for real-time performance. Though the results show that UAVMOT achieves state-of-the-art performance in terms of accuracy and efficiency for the given task, it has some drawbacks like occlusion handling and robustness to extreme weather conditions.The proposed DQNdot framework uses YOLOv8 for object detection, UAVMOT for Feature Extraction and Association, Kalman Filtering for Motion Prediction and Deep Q-Network (DQN) for Decision-Making.This framework pursues objects even in states of partial or complete obscurity. DQNdot can not survive more complex areas like seashores,because it only focuses on road side surveillance.

2.1 Overall Inferences

From the above discussion and Table 1, the following challenges and limitations are inferred.

a. When tracking multiple objects, occlusions can occur, where one object hides or partially obscures another. Dealing with occlusions is a complex problem in multi-object tracking.

Table 1 Comparison of various existing algorithms for object detection and tracking.

Author[Year]	Techniques	Dataset	Advantages	Limitations
Roberto Opromolla et.al. [2019]	YOLOv2	Own Dataset	High accuracy	Identifying targets at far range below the horizon
Shivani Kapania et.al [2020]	YOLOv3, deepSORT, RetinaNet	MARS	Improved accuracy in detecting objects from considerable heights	Finding a balance between the accuracy and the speed at which the tracking algorithm operates in real-time scenarios.
Zhengwei Zhu et.al. [2021]	DDDQN	-	convergence speed and convergence stability	Developing the motion model for the USV in challenging marine environments
Li-Yu Lo et.al. [2021]	YOLOv4 Tiny, DQN	Customized dataset	Does not require the prior knowledge of the environment	Not able to predict the target movement,collision due to static and dynamic objects
Chandra Has Singh et.al. [2022]	F-RCNN	VisDrone 2019	Identifies small objects	Training complexity
Jin-Hyeok Park et.al. [2022]	DQN, deep-SORT	VisDrone 2019, OTB-100	High precision	Real time testing is needed, Robustness to extreme weather condition
Li, Y et.al. [2022]	YOLOv5s, Ghostnet, Transformers	MariDrone	Lesser computational cost	Handling adverse weather, cluttered backdrops
Hy Nguyen et.al. [2023]	DQN, Stack of frames, segmentation, depth images	-	-	Suitable only for multi rotor UAVs
Shuai Liu et.al. [2023]	UAVMOT	VisDrone2019, UAVDT	Increased object feature association	Hardware requirements, Robustness to extreme weather condition, Processing speed
Neethu Subash et.al. [2023]	YOLOv8, DQN	UAVDT, VisDrone	Handles occlusions	Not suitable for complex areas like seashores

b. UAVs often operate in cluttered environments with many objects of interest and distractions, making it challenging to differentiate and track specific objects accurately. Changes in lighting conditions during UAV flights can affect object appearance, making it harder to track objects accurately.

c. In multi-object tracking, maintaining the identity of objects over time is essential. However, re-identifying objects correctly after they are temporarily lost or occluded can be difficult, especially if there are similar-looking objects in the scene.

d. UAVs often have limited onboard computational resources. Real-time processing and tracking multiple objects can be computationally expensive, which may impact the system's overall performance.

2.2 Contributions

a. The proposed framework combines YOLOv8 with Nano instance segmentation to form a powerful framework for object detection and tracking. YOLOv8's efficiency in precise bounding box detection is enhanced by Nano instance segmentation, which provides accurate mask predictions within these bounding boxes. This synergy ensures high accuracy, fast processing, and adaptability to diverse scenarios, making it well-suited for real-time applications.

b. In addition to this fusion, the proposed framework integrates DDDQN to further enhance object tracking capabilities, particularly in dynamic environments with multiple moving objects. DDDQN's adaptive decision-making, facilitated by the separation of value and advantage functions, improves resilience in object tracking.

c. With the proposed fusion of YOLOv8, Nano Instance Segmentation and DDDQN, the proposed occlusion-resistant tracking framework handles any difficult situations and continue tracking objects even when they are temporarily hidden from view.

d. A thorough analysis of proposed work along with various existing frameworks is done with UAVDT and VisDrone datasets, considering metrics such as MOTA, MOTP, IDF1 Score, Inference Time, and FPS. The proposed framework demonstrated superior performance in terms of accuracy, adaptability, and efficiency, highlighting its potential for real-world applications.

3 Proposed Nano Instance Segmentation with DDDQN (NISD) framework

The proposed algorithm combines different object within the bounding boxes. This phase helps us to get the abstract representation of the entities in the frame by doing feature extraction, bound box, region segmentation , nano level segmenatation, object representation and identification. Phase 2 of the proposed work uses the Deep Dueling Dueling Q-Network (DDDQN) algorithm. Dueling architecture, in this phase, plays a pivotal role in optimizing the decision-making process of the agent, enabling it to predict object locations and trajectories effectively. This phase analayses and predicts the target's movement and ensures tracking to the finest state using the DDDQN's Dueling architecture.As it has been assisted by yolov8 nanosegmentation, the tracking of the object can be more efficiently done along with the occlusion avoidance , overlapping and obstacle avoidance.

In the following subsections, all these phases in the proposed framework are discussed in detail.

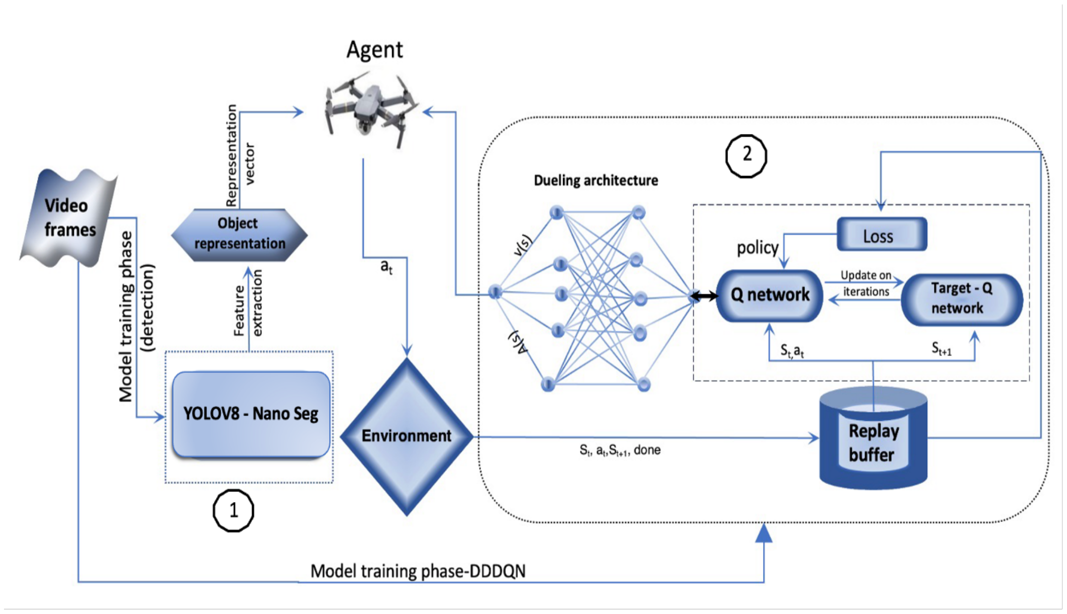

Fig 1 Proposed Framework.

3.1 Phase 1: YOLOv8 and Nano Instance Segmentation(NIS)

You Only Look Once(YOLOv8) is a state-of-the-art object detection algorithm that has gained popularity due to its real-time performance and high accuracy. Nano-Instance Segmentation is a new approach that extends object identification to the level of individual instances, allowing for exact segmentation of objects even in cluttered environments. The combination of YOLOv8 with Nano-Instance Segmentation(NIS) provides a strong tool for applications like object tracking, anomaly detection, and autonomous navigation.

3.1.1 YOLOv8

The YOLOv8 method sets out to solve object identification in the vast field of UAV-captured video frames by carefully examining each frame in order to determine object locations. YOLOv8 splits the image using a grid-based method, making predictions about the locations of bounding boxes, classifying the labels, and calculating item confidence scores. Using a YOLOv8 model that has already been trained, the proposed model locates desired target objects that are hidden in the video frames captured by UAVs. YOLOv8 can be represented as a function in 1 that maps an input image or video frame to a set of bounding boxes and class probabilities as

$$Y = fYOLOV8(X)\,, \tag{1}$$

where X is the input image or video frame, Y is the set of bounding boxes and class probabilities and fYOLOV8(X) is the YOLOv8 function. It utilizes a learned framework to anticipate the arrangement of enclosing structures, allocate class titles, and ascertain object trust scores.

3.1.2 Segmentation Map

Segmentation Map is used in multi-object tracking to improve the accuracy of object detection and tracking by providing additional information about the objects' boundaries and shapes. Specifically, segmentation can help to distinguish between overlapping objects, handle occlusions, and improve the localization accuracy of objects. The segmentation network function is represented as

$$S = fSeg(X)\,, \tag{2}$$

Where S is the segmentation map and *fseg* is the segmentation network function applied to the input image X. To achieve the best object detection we combine both functions Y and S. Among the different segmentation models, the proposed model utilizes Nano instance segmentation that is lightweight and efficient, making it perfect for real-time applications. This is accomplished by employing a smaller model and a simpler algorithm than typical instance segmentation approaches. The proposed framework uses NIS which shows better accuracy and interpretability when compared to Semantic and Instance segmentation as shown in table 2,[12],[19].[20] The traditional object detection methods lack precision and clarity, more specifically, semantic segmentation methods prioritize precision over interpretability. Whereas, the nano instance segmentation offers a harmonious blend of both. The segmentation map, generated by the segmentation network function (fSeg), offers a detailed insights into object boundaries and shapes in multi-object tracking. This information is invaluable for distinguishing overlapping objects, handling occlusions, and improving the accuracy of object localization. When integrating this segmentation data (S) with object detection results (Y), a fusion process is applied. This fusion involves techniques such as weighted averaging or element-wise multiplication, tailored to the specific application's requirements, ensuring a seamless integration of both sets of information. Further, this nano instance segmentation

Table 2 Comparison of various segmentation methods.

Algorithm	Accuracy	Interpretability
Nano Instance segmentation	High	High
Semantic Segmentation	High	Low
Instance Segmentation	High	Medium

algorithm is tested against with Mask R-CNN and Single-Shot multibox Detector(SSD) for two data sets UAVDT and VISDRONE and this comparison is given in Table 3. The nano instance segmentation achieves an accuracy of 85.60% while zipping through the processing in a mere 100ms. Although Mask R-CNN consumes 200 ms of processing power, it achieves an accuracy of 82.50%. Real-time applications face significant problems due to its extended inference periods, which are 300 ms for the UAVDT dataset and 600 ms for the VISDRONE dataset, respectively. While maintaining a moderate accuracy of 70.00%, the SSD method smoothly navigates across both datasets in 150 ms and 200 ms, respectively, striking a careful balance between computation and inference times. From this analysis, it is inferred that the SSD delivers efficiency for real-time settings, Mask R-CNN provides precision at the cost of longer inference periods, while the nano instance segmentation thrives in both accuracy and speed and becomes a sensible option where the processing in real time is critical.

Table 3 Comparison of mask RCNN and SSD with NIS method.

Algorithm	Accuracy	Computation time (ms)	Inference time (UAVDT)	Inference time (VISDRONE)
NIS	85.60%	100	50	100
Mask R-CNN	82.50%	200	300	600
SSD	70.00%	150	200	200

3.1.3 Exploring the benefits of Phase 1 in the proposed framework

The heat map presented in figure 2 illustrates the significant differences in performance between YOLOv8 and Phase 1 of the proposed model which is the combination of YOLOv8 and nano instance segmentation. Though there are noticeable cases of false negatives, YOLOv8 demonstrates proficiency in detecting true positives. On the contrary, YOLOv8 with the nano instance segmentation in phase 1 of the proposed model demonstrates unparalleled accuracy in identifying real positives and minimising false negatives. Furthermore, as compared to YOLOv8, it exhibits a lower false positive rate, demonstrating its effectiveness in improving detection. This increases the accuracy and dependability, making it a robust and meticulous choice for precision-driven detection projects.

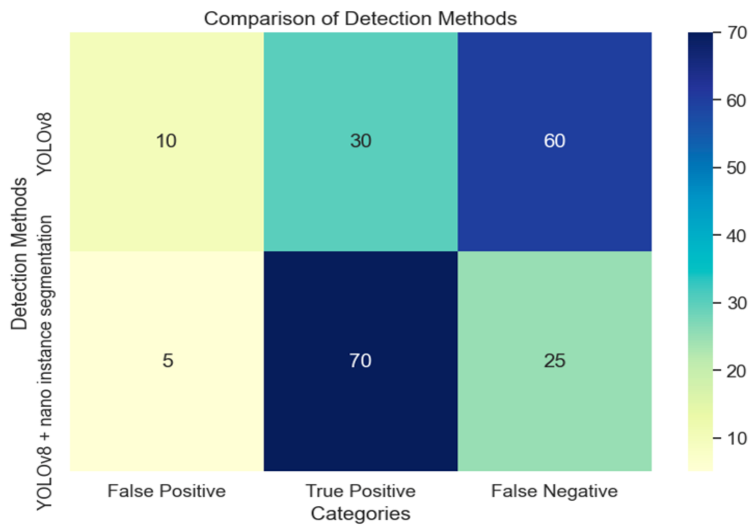

Fig 2 YOLOv8 Vs Phase 1 of the proposed model.

3.2 Phase 2: Dueling Architecture with DDQN(DDDQN)

The Dueling Architecture combines the advantages of both the Double DQN and Dueling DQN algorithms. It uses two separate Q-value streams to estimate the value of each action and the value of the state. This allows the algorithm to better handle overestimation of Q-values and to learn the value of each state more efficiently. The figure 3 shows the dueling architecture in the pro-

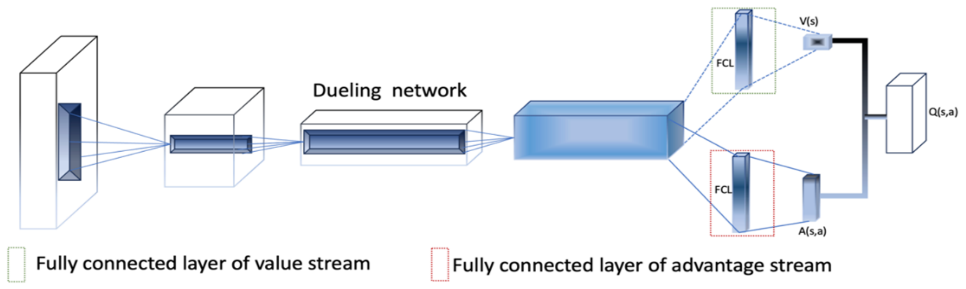
Dueling network
FCL
V(s)
FCL
A(s,a)
Q(s,a)
Fully connected layer of value stream
Fully connected layer of advantage stream

Fig 3 Dueling Architecture.

posed model and it consists of main two streams namely advantage and value streams. Based on these streams, Q value is decided and actions will be performed. Dueling architecture addresses a common problem in Q-Learning where the state- value function with a large number of actions becomes difficult to estimate correctly. Moreover, it aims to improve stability and convergence speed of the Q learning algorithm.The Dueling architecture separates the Q-values into two components: the value stream function (V) and the advantage function (A).

3.2.1 Value and Advantage Functions in DDDQN

The DDDQN (Dueling Double Deep Q-Network) architecture involves two crucial components: the Value function (V) and the Advantage function (A). These functions play a key role in approximating the Q-values for state-action pairs. The combined loss function, denoted as $L(\theta)$, encompasses both the Value and Advantage functions as given below:

$$L(\theta) = \frac{1}{\text{batch\_size}} \sum_j (y_j - Q(s_j, a_j; \theta))^2 \tag{3}$$

Here, the loss is computed for each sample in the batch and is the mean squared difference between the predicted Q-value ($Q(s_j, a_j; \theta)$) and the target value (y_j). The loss is then minimized during training. These are given as in the following equations 4 and 5.

- **Target Value (y_j):** It represents the expected cumulative reward for taking action a_j in state s_j. It is calculated using the Bellman equation

$$y_j = r_j + \gamma \left(V(s_{j+1}; \theta_{\text{target}}) + A(s_{j+1}, \arg\max_{a'} A(s_{j+1}, a'; \theta)) \right) \,, \tag{4}$$

  where r_j is the immediate reward, γ is the discount factor, $V(s_{j+1}; \theta_{\text{target}})$ is the value function for the next state, and $A(s_{j+1}, \arg\max_{a'} A(s_{j+1}, a'; \theta))$ is the advantage function for the best action in the next state.

- **Q-value Prediction ($Q(s_j, a_j; \theta)$):** Represents the predicted Q-value for taking action a_j

in state s_j with the current Q-network parameters θ. It combines the Value and Advantage functions as

$$Q(s_j, a_j; \theta) = V(s_j; \theta_V) + \left(A(s_j, a_j; \theta_A) - \frac{1}{\text{no.of actions}} \sum_{a'} A(s_j, a'; \theta_A) \right) , \tag{5}$$

where θ_V and θ_A are the weight vectors for the Value and Advantage streams, respectively.

The goal during training is to adjust the Q-network parameters to minimize this loss, improving the accuracy of predicting Q-values for optimal decision-making.

3.2.2 Reward function

In the dueling architecture, the Q-value for each action a in state s is calculated as the sum of the value function and the advantage function minus the mean of the advantage function. In DQN, DDQN, and DDDQN, the Q value is estimated using a Q network, which is a neural network trained to predict the Q values for a given state-action pair. The Q value is a measure of how good it is to take a particular action in each state. In DDDQN, the Q value is calculated using the following dueling architecture

$$Q(s, a; \theta) = V(s; \theta_V) + \left(A(s, a; \theta_A) - \frac{1}{\text{no.of actions}} \sum_{a'} A(s, a'; \theta_A) \right) . \tag{6}$$

Here, $Q(s, a; \theta)$ represents the Q-value for the state-action pair, $V(s; \theta_V)$ is the value function, and $A(s, a; \theta_A)$ is the advantage function. The term $\frac{1}{\text{no.of actions}} \sum_{a'} A(s, a'; \theta_A)$ represents the mean of the advantage function. But this is in contrast to the Q-learning Bellman equation as given in equation 7.

$$Q(s,a) = E[R + \gamma \cdot \max_{a} Q(s', a')] \,, \tag{7}$$

where $Q(s,a)$ is the Q value for the state-action pair, R is the reward received for taking action a in state s, γ is the discount factor, s' is the next state, and a' is the action taken in state s'. In DQN and DDQN, the reward function is used to train the Q network. The Q network learns to predict the expected reward for taking a particular action in a given state. The reward function is therefore essential for these algorithms to learn and to take actions that lead to a high reward. This means that the reward function may encourage the agent to take actions that do not lead to a high reward. However, DDDQN need not worry about the reward function in the same way as DQN and DDQN. It uses a dueling architecture, which separates the value function and the advantage function. The advantage function is a measure of how much better one action is than another in a given state and which is given the following equation 8.

$$\text{Reward}(s,a) = V(s;\theta_V) + A(s,a;\theta_A) \,, \tag{8}$$

Where θ_V and θ_A denote the weights of the value stream and advantage stream, respectively. The value function (V) estimates the overall expected reward for a given state, while the advantage function (A) quantifies how much better or worse each action is in comparison to the average action within that state. The combination of these components in the reward calculation allows DDDQN to make well-informed decisions, taking into account both the global state value and the relative advantage of each action within that state.

This approach enables the DDDQN algorithm to learn optimal actions that are more likely to result in a high reward, even in scenarios where a traditional reward function may not be well-defined. By emphasizing the comparison between actions rather than relying on an absolute reward value, DDDQN proves to be less susceptible to biases in the reward function. Consequently, the algorithm doesn't necessitate the explicit configuration of a reward function, showcasing its adaptability in learning actions that lead to favorable outcomes.

Algorithm 1 Proposed NISD framework.

```
1: **Initialize** replay buffer $\mathcal{D}$ with capacity $replay\_buffer\_capacity$,$\theta_{\text{target}} = \theta$,
2: **Initialize** Q-network $\theta$ with Dueling Architecture, $\epsilon = \epsilon_{\text{initial}}$    ▷ *Initialization*
3: **for each** episode **from** 1 to *num_episodes* **do**
4:     **Initialize** state $s_1$,
5:     **Reset** episode total reward $R = 0$
6:     **for each** time step $t$ **from** 1 to $T$ **do**
7:         **With probability** $\epsilon$, select a random action $a_t$    ▷ *Explore vs. Exploit*
8:         **Otherwise**, select $a_t = \arg\max_a (V(s_t;\theta) + A(s_t, a;\theta))$
9:         **Execute action** $a_t$ **and observe reward** $r_t$, **next state** $s_{t+1}$
10:        $R = R + r_t$
11:        **Store transition** $(s_t, a_t, r_t, s_{t+1})$ **in replay buffer** $\mathcal{D}$
12:        **Sample minibatch** $(s_j, a_j, r_j, s_{j+1})$ **from replay buffer** $\mathcal{D}$
13:        **Set target** $y_j = r_j + \gamma \left(V(s_{j+1};\theta_{\text{target}}) + A\left(s_{j+1}, \arg\max_{a'} A(s_{j+1}, a';\theta)\right)\right)$    ▷ *Bellman equation*
14:        **Update value & advantage function** $V$ **by minimizing the loss**: $L(\theta) = \frac{1}{\text{batch\_size}} \sum_j \left(y_j - Q(s_j, a_j;\theta)\right)^2$
15:        **if** $t \bmod \text{target\_update\_frequency} = 0$ **then**
16:            **Update target Q-network weights**: $\theta_{\text{target}} = \theta$
17:        **end if**
18:        **Detect objects in** $s_{t+1}$ **using** YOLOv8:
19:        $Y = f_{\text{YOLOv8}}(s_{t+1})$    ▷ *Mapping X to bounding boxes & class probabilities*
20:        **Segment objects in** $s_{t+1}$ **using** Nano instance segmentation:
21:        $S = f_{\text{Seg}}(s_{t+1})$    ▷ *Mapping an X to a segmentation map*
22:        **Update** $s_{t+1}$ **to include object detection & segmentation results**:
23:        $s_{t+1} = \text{Combine}(Y, S)$    ▷ *Integration of YOLOv8 & NIS*
24:        **Calculate reward** $r_t$ using the custom reward function for DDDQN:
25:        $r_t = \text{CalculateReward}(\theta_V, \theta_A, s_{t+1})$    ▷ *Custom reward function*
26:    **end for**
27: **end for**
28: **if** episode is done **then**
29:     **break**
30: **end if**
```

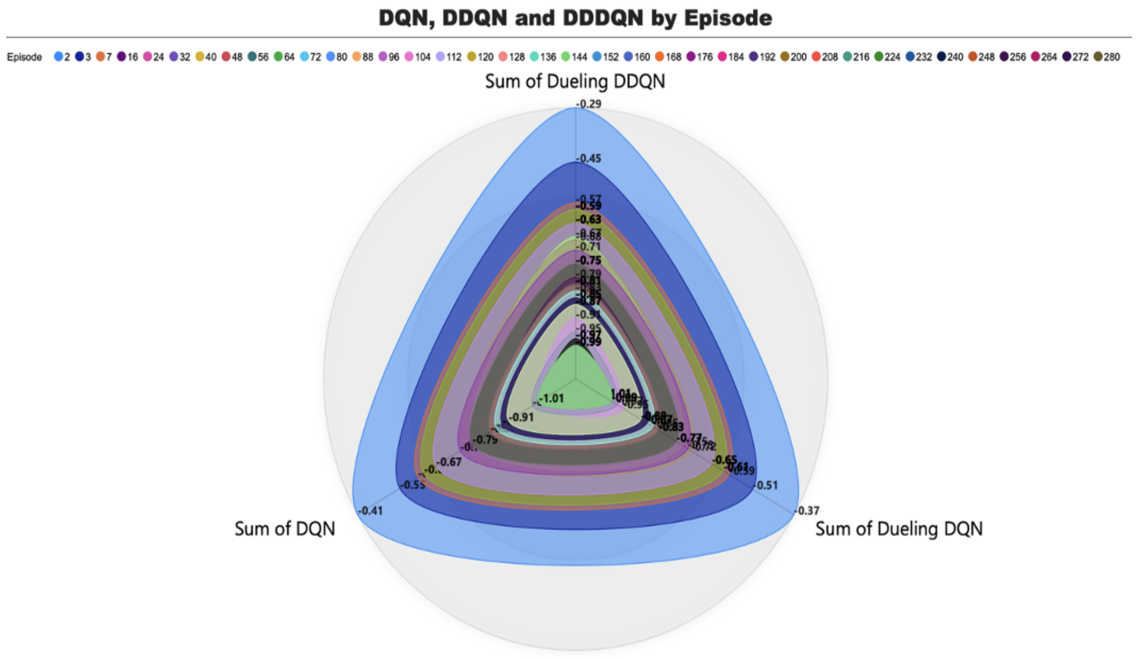

Fig 4 Rewards obtained for 500 episodes (calculated from episode 0 to 500).

Figure 4 shows the rewards obtained over episodes from 0th episode to 500th episode. It shows the performance and reward configured on the model training phase of DQN, DDQN, DDDQN and made a comparison over it. It is proved that compared to DQN and DDQN, the proposed model with DDDQN performs well and achieves better rewards than the others. The complete algorithm for the proposed model is given in Algorithm 1.

3.2.3 Object Detection Confidence and Tracking Confidence

For YOLOv8 and NIS in the proposed model, the confidence scores are computed using equations 9 and 10:

$$C_i = P(\text{class}_i) \cdot P(\text{object}_i) \quad \text{(Object Detection Confidence)} \tag{9}$$

$$Q_i = \frac{Q(s,a)}{\max_{a'} Q(s,a')} \quad \text{(Tracking Confidence)} \tag{10}$$

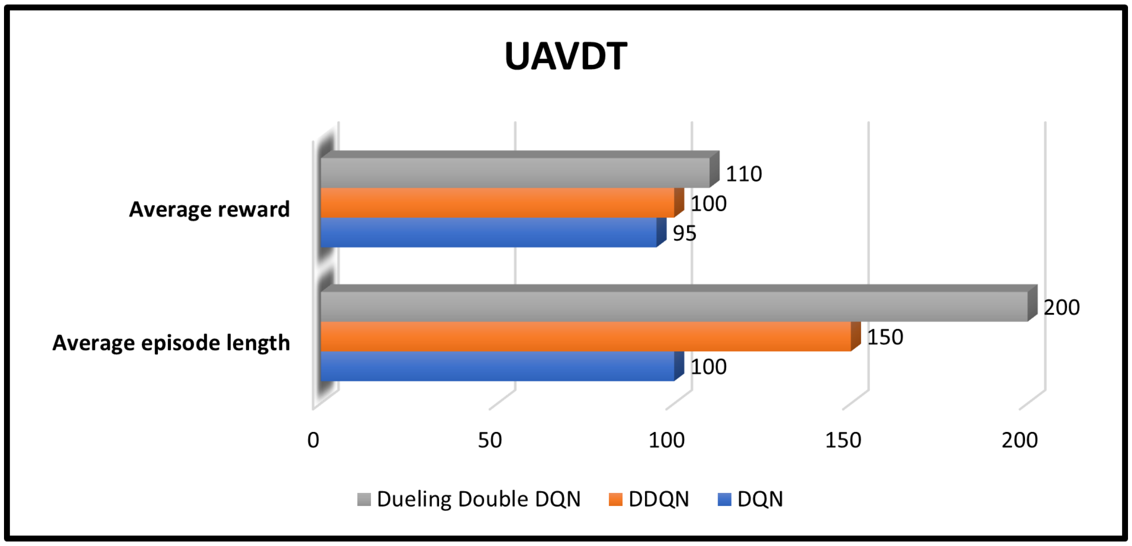

Fig 5 Average reward and average episode length obtained while training DQN, DDQN, DDDQN in UAVDT dataset.

C_i represents the confidence score for each detected object, and Q_i is the confidence score associated with the tracked objects based on DDDQN as shown in Algorithm 2. To merge the information from YOLOv8, NIS and DDDQN, a fusion weighted score (F_i) is calculated as in Equation 11.

$$F_i = \alpha \cdot C_i + (1 - \alpha) \cdot Q_i \tag{11}$$

Here, α serves as a parameter controlling the balance between the contributions of object detection and tracking. Adjusting α allows fine-tuning the system's sensitivity to the obtained results from the proposed model. The decision on whether to accept or discard an object is determined by the fusion weighted score F_i. If F_i exceeds a specified threshold, the object is accepted; otherwise, it is discarded. This threshold can be adjusted based on the desired trade-off between precision and recall. Algorithm 2 outlines the fusion process, iterating through detected and tracked objects to compute C_i, Q_i, and ultimately F_i.

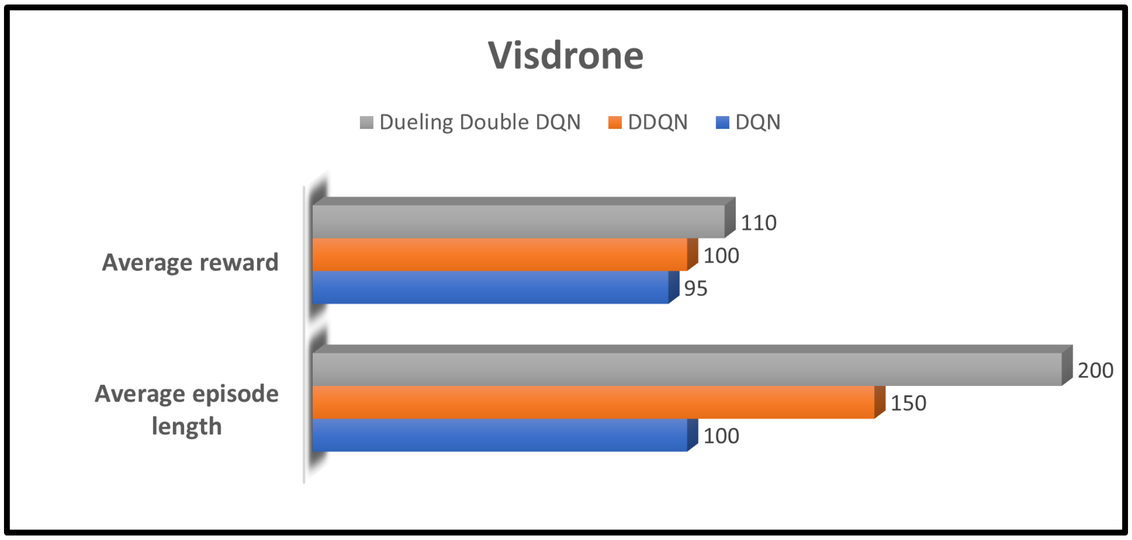

Fig 6 Average reward and average episode length obtained while training DQN, DDQN, DDDQN in VISDRONE dataset.

Algorithm 2 Fusion Algorithm

```
1: for $i \leftarrow 1$ to $N$ do
2:     Compute $C_i$ using YOLOv8
3:     Compute $Q_i$ using DDDQN
4:     Compute $F_i$ using the fusion weighted score
5:     if $F_i >$ threshold then
6:         Accept object $i$
7:     else
8:         Discard object $i$
9:     end if
10: end for
```

To summarize, the proposed model with the clever combination of DDDQN, YOLOv8 object detection, and Nano instance segmentation, is carefully tailored for multi-object tracking in UAV scenarios. By using real-time object recognition and segmentation processing, the integration of YOLOv8 with Nano instance segmentation enhances perception capabilities and brings in convolution. Duelling DDQN integration provides a more sophisticated understanding of actions by separating the value and advantage functions. The performance of this proposed model is analyzed in the next section.

4 Simulation and Performance Analysis

In the UAVDT and VisDrone datasets, the performance of the proposed model, along with other existing models, is examined. Subsection 4.1 compares the object tracking capability of these models. In Section 4.2, the loss function used for training the proposed model is elaborated, and the loss values of different frameworks are discussed. Subsection 4.3 analyzes other metrics such as epsilon, discount factors, and accuracy of the models with varying numbers of episodes. All these discussions are based on the simulation parameters as given in Table 4.

Table 4 Simulation Parameters and their value.

Parameter	Value	Parameter	Value
Number of objects to track	10	Object size	100 x 100 pixels
Object speed	10 pixels/frame	Object occlusion	20%
Environment complexity	high	Simulation time	100 frames
YOLOv8 layers	80	YOLOv8 learning rate	0.001
NIS layers	16	NIS learning rate	0.0001
DDDQN hidden layers	3	DDDQN learning rate	0.0001
Learning decay rate	0.99	Epsilon	0.1
Batch size	128	Optimizer	Adam
Replay buffer size	100000	Alpha	0.001

4.1 Comparison Metrics for Object Tracking

The following metrics are measured to analyze the object tracking capability of the proposed model and other frameworks. The existing frameworks in [4-13] are compared with the proposed model and the inferences are also discussed in this subsection.

i. **Multiple Object Tracking Accuracy (MOTA):** MOTA takes into account several factors, including the number of missed detections, false positives, identity switches, and fragmentations in the tracked objects. The single and multiple agent MOTA scores are given as

$$MOTA(SingleAgent) = 1 - \frac{M_m + F_p + M_i}{G},$$

where M_m is the number of missed detections, F_p is the number of false positives, M_i is the number of identity switches, and G is the total number of ground truth.

$$MOTA(MultipleAgents) = 1 - \frac{\sum M_m + \sum F_p + \sum M_i}{\sum G},$$

where $\sum M_m$, $\sum F_p$, and $\sum M_i$ represent the sum of missed detections, false positives, and identity switches across all agents, respectively.

ii. **Multiple Object Tracking Precision (MOTP):** The MOTP metric calculates the average overlap between the predicted bounding boxes (detections) and their corresponding ground truth bounding boxes as

$$\text{MOTP} = 1 - \frac{\sum_{t=1}^{GT} \min_{t' \neq t} d_{tt'}}{GT},$$

where $d_{tt'}$ is the distance between the ground truth bounding box for track t and the predicted bounding box for track t'. GT is the total number of ground truth tracks in the video. MOTP provides a measure of the accuracy of object localization by calculating the average overlap across all tracks. A higher MOTP indicates better alignment between predicted and ground truth bounding boxes, reflecting improved tracking precision.

iii. **IDF1 Score:** The IDF1 score is a metric that combines precision and recall to provide a comprehensive measure of tracking performance.

$$\text{IDF1 score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}},$$

where Precision is the ratio of true positives to the total number of predictions, and Recall is the ratio of true positives to the total number of ground truth objects. The IDF1 score considers both false positives and false negatives, providing a balanced assessment of a tracking system's ability to correctly identify and follow objects over time. A higher IDF1 score indicates a more accurate and reliable tracking system.

iv. **Inference Time:** Inference time refers to the duration it takes for the object tracking model to process input data and produce an output or prediction. It is a crucial metric for real-time applications, as shorter inference times imply faster processing and, consequently, quicker response in dynamic environments. Lower inference times are desirable for ensuring timely and efficient object tracking.

v. **Frames per Second (FPS):** Frames per Second is a measure of how many frames an object tracking model can process in one second. It indicates the speed at which the model can analyze and track objects in a video sequence. Higher FPS values are preferred in applications where real-time tracking is essential, ensuring that the model can keep up with the pace of the input video. A higher FPS value contributes to smoother and more responsive object tracking in dynamic scenarios.

The proposed framework (the fusion of DDDQN, YOLOV8, and nano instance segmentation) has demonstrated superior MOTA, MOTP, and IDF1 scores compared to all frameworks with

Table 5 Performance Comparison with UAVDT dataset.

Framework	MOTA (single target)	MOTA (Multiple target)	MOTP	IDF1 score	Inference time	FPS
Proposed NISD Model	92.5	86.7	83.7	79	150	6.67
LDVRL[17]	89.2	84.7	83	78.3	200	5
DQN + deepSORT[15]	90.9	86.3	83.5	78.8	100	10
DQN + YOLOv4tiny[13]	89.9	85.3	82.9	78.2	100	10
DDDQN + YOLOv3 + DeepLab[8]	91.1	86.5	83.6	78.9	175	5.71
DDQN + SSD + DeepSort[921]	90.8	86.2	83.4	78.7	200	5
DQN + UAVMOT + YOLOV8[18]	92.2	84.7	83	78.3	125	8
GGTYOLO + DQN[16]	90.4	85.7	83.2	78.5	100	10
Fast RCNN + MOT[14]	87.6	83.1	81.4	76.6	250	4
DDDQN + VMP algo[22]	91.4	86.4	80.3	77.5	200	7

UAVDT and VisDrone datasets as listed in Table 5 and 6 respectively. The pictorial comparison is also depicted in Figs 8 and 9. While maintaining a moderate inference time and Frames per Second (FPS), this fusion showcases notable performance in the realm of computer vision tasks. In figure 7, the different frameworks are compared based on MOTA score. Even with a little reduced MOTA, the framework *LDVRL*[17] maintains the balance between MOTA and MOTP values. The fusion of DQN with Deepsort[15] and DQN with YOLOV4-tiny[13] frameworks achieve a good balance between real-time processing speed and accuracy, achieving respectable MOTA scores with 100ms inference times and 10FPS. With an inference time of 300 ms and a frame rate of 3.33 FPS, the DDQN framework with SSD and DeepSort[15] performs well in situations where a lower frame rate is enough. The accuracy and precision of DQN with UAVMOT and YOLOV8[18]are similar to those of DQN with YOLOV4-tiny,[13] demonstrating a variety of detection models with consistent tracking performance. With a 10 FPS and a 10 ms inference time, the combination of DQN

Table 6 Performance Comparison with vis-drone dataset.

Framework	MOTA (single target)	MOTA (Multiple target)	MOTP	IDF1 score	Inference time	FPS
Proposed NISD Model	86.5	81.7	80.2	75.6	175	5.71
LDVRL[17]	84.3	79.6	78.6	74	200	5
DQN + deepSORT[15]	85.8	81	79.4	74.8	125	8
DQN + YOLOv4tiny[13]	84.6	79.8	79	74.4	100	10
DQ + YOLOv3 + DeepLab[19]	86.3	81.5	80	75.4	250	4
DDQN + SSD + DeepSort[9,21]	86	81.2	79.8	75.2	300	3
DQN + UAVMOT + YOLOV8[18]	84.6	79.8	79	74.4	150	6.67
GGTYOLO + DQN[16]	85.7	80.9	79.7	75.1	100	10
Fast RCNN + MOT[14]	83.2	78.5	77.6	72.9	250	4
DDDQN + VMP algo[22]	86.0	80.5	79.2	74.9	250	7

and GGTYOLO[16] demonstrates optimised processing and represents effective object tracking with strong MOTA and MOTP results.

When compared to other frameworks, the combination of Fast RCNN and MOT[14] framework exhibits lower accuracy and precision, which may indicate limitations when it comes to object tracking tasks. Although the framework DDDQN with VMP[22] shows good MOTA and MOTA (many targets) scores, it lacks in IDF1 score and precision, indicating the need for improvement in object trajectory management.

4.2 Loss Function

To train the DDDQN in the proposed model and DDQN, the dueling architecture is utilized, incorporating the Mean Squared Error (MSE) between the target Q-value and the predicted Q-value.

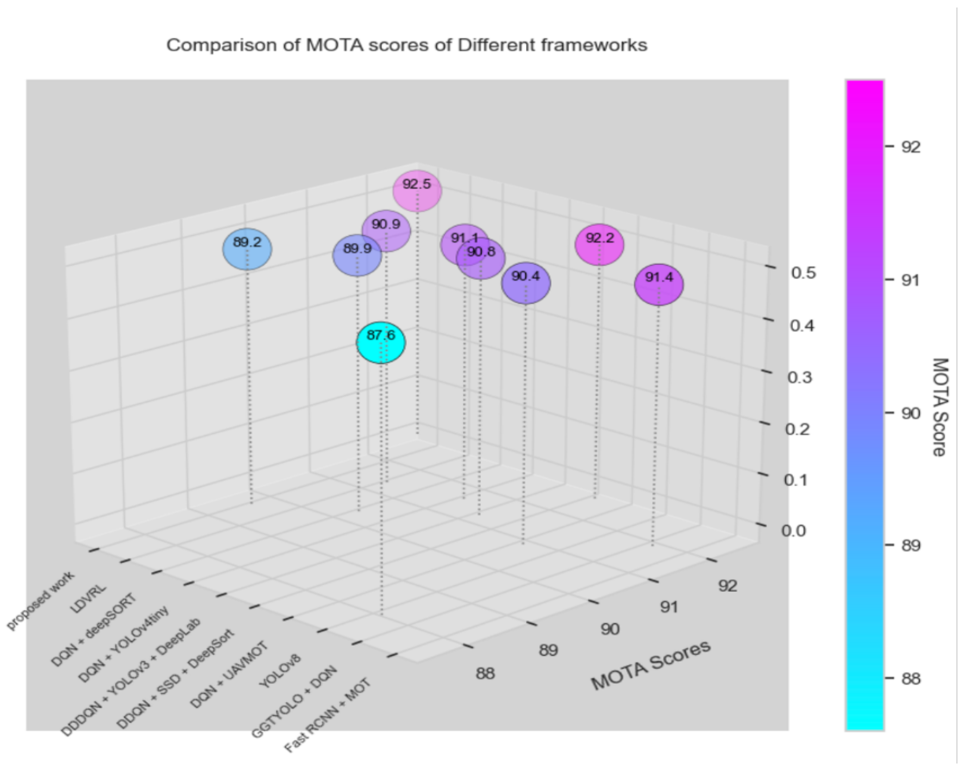

Fig 7 Comparison of MOTA of different frameworks.

However, the standard DQN does not undergo dueling architecture.

$$DDDQN(L) = \left[r + \gamma \cdot \left(\max_{a'} Q(s', a'; \theta^-) - V(s', \theta^-)\right) - Q(s, a)\right]^2$$

$$DDQN(L) = \left[r + \gamma \cdot \max_{a'} Q(s', a'; \theta^-) - Q(s, a)\right]^2$$

$$DQN(L) = \left[r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)\right]^2$$

where L is the loss, r is the immediate reward after taking action a in state s, γ is the discount factor , $Q(s, a)$ is the predicted Q-value , s' is the next state, $\max_{a'} Q(s', a'; \theta^-)$ is the maximum Q-value θ^-, and $V(s', \theta^-)$ is the value of state s' using the target network parameters θ^-.

$$Q\_target(s, a) = r + \gamma \cdot \max_{a'} Q\_target(s', a')$$

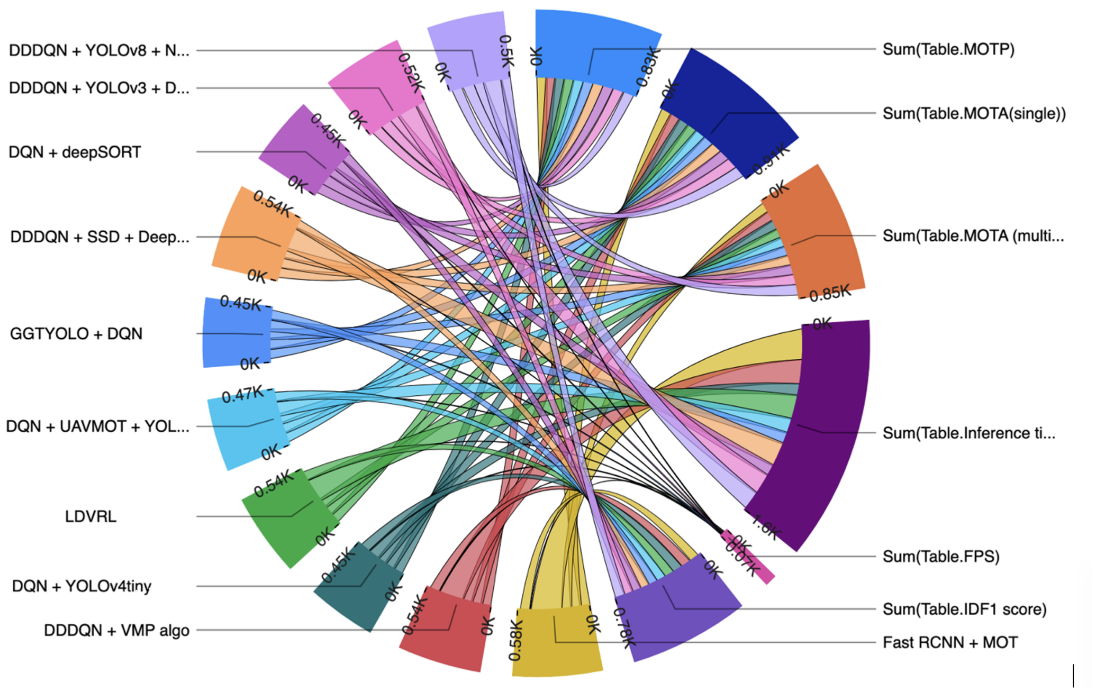

Fig 8 Performance Comparison of frameworks [4-13] on UAVDT Dataset.

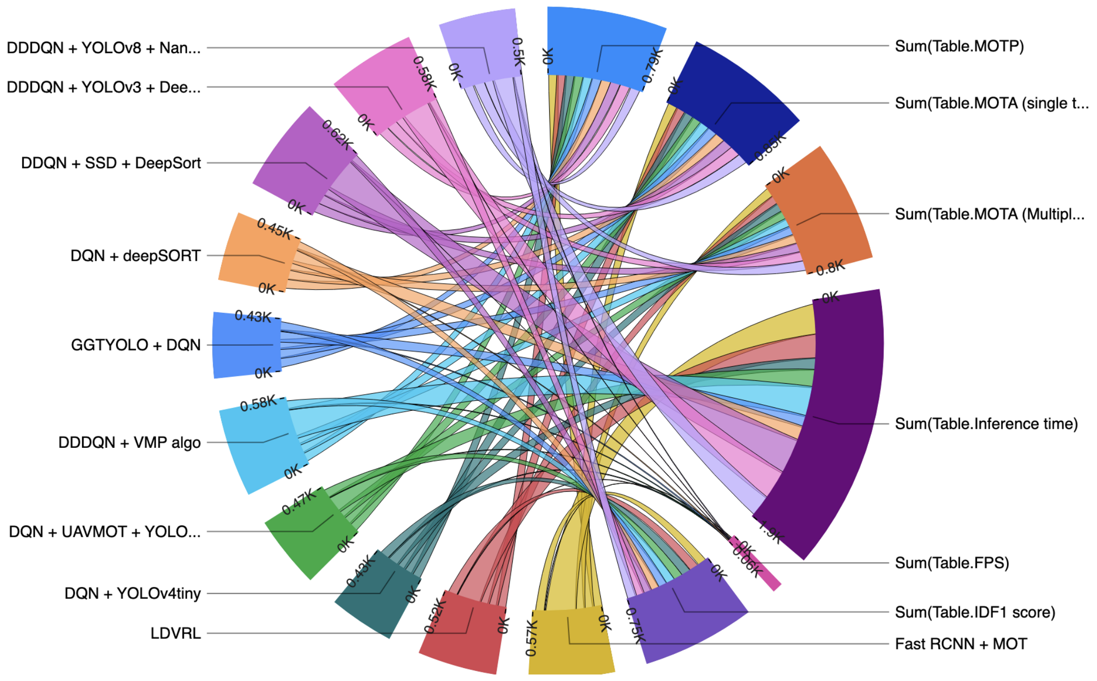

Fig 9 Performance Comparison of frameworks [4-13] on VisDrone Dataset.

The target Q-value (Q_target) is updated using the Bellman equation. In this equation, r denotes the immediate reward after taking action a in state s, s' is the next state, γ is the discount factor, and $\max_{a'} Q\_target(s', a')$ is the maximum Q-value in the next state. The Q-value is computed using the Temporal Difference (TD) approach.

4.3 *Epsilon and discount factor*

As seen in Table 8, the UAVDT dataset reveals that low epsilon values (0.1) consistently yield higher accuracy levels (90%, 85%, and 80%). This indicates a focused investigation on the part of the agent, leading to improved learning results. High discount factors (0.99 and 0.999), however, give the datasets lower loss values, highlighting the need of giving future rewards priority for convergence. However, increasing the number of episodes for UAVDT leads to decreasing accuracy and incentives, suggesting the difficulties in maintaining optimal performance over long durations.

Table 7 Loss obtained for DQN, DDQN, DDDQN in the Proposed model.

Framework	Learning rate	Batch size	Number of epochs	Loss
DDDQN in proposed model[12]	0.0001	32	100	0.001
DQN[15]	0.0001	32	100	0.002
DDQN[9]	0.0001	32	100	0.0015

Similar trends are observed in the VisDrone dataset, where agent performance (95%, 90%, and 85%) is enhanced by decreasing epsilon values. These datasets show that higher discount factors are associated with lower loss values, indicating that the agent is more likely to predict future rewards. This analysis emphasises how important it is to carefully adjust hyperparameters in order to obtain the best possible agent behaviour over a range of datasets and episode lengths. Lower epsilon levels combined with appropriate discount factors seem to be especially effective for the UAVDT dataset.

Table 8 Epsilon, Discount factor, loss, Accuracy, Reward and Episodes of UAVDT and VisDrone dataset.

Dataset	Epsilon	Discount factor	Loss	Accuracy	Reward	Episodes
UAVDT	0.1	0.9	0.001	90%	95	100
UAVDT	0.1	0.99	0.002	85%	85	150
UAVDT	0.1	0.999	0.003	80%	75	200
VisDrone	0.1	0.9	0.001	95%	110	100
VisDrone	0.1	0.99	0.002	90%	100	150
VisDrone	0.1	0.999	0.003	85%	90	200

5 Conclusion

This paper introduces a robust methodology for object detection in Unmanned Aerial Vehicles (UAVs) using a fusion of YOLOv8x, Nano instance segmentation, and the DDDQN algorithm. The proposed model overcomes inherent limitations of traditional approaches and showcases superior performance.The integration of YOLOv8x and the DDDQN algorithm addresses obstacles, variations in object size, and unexpected movements, making it particularly adept at object detection in densely populated environments.The effectiveness of the proposed algorithm is further supported by simulation results, which show improved accuracy of 86.7 and 81.7 on the UAVDT and vis-drone datasets, respectively, and IDF1 scores of 79 and 75.6 on the UAVDT and vis-drone datasets, respectively, when compared to a variety of other object detection algorithms. The proposed methodology holds promise for practical implementation, offering a robust solution for object detection in challenging environments. The integration of Internet of Drones (IOD) technologies, utilizing networked UAVs for collaborative and synchronized object detection tasks, emerges as a potential focus for our future research.

References

1 A. Ramachandran and A. K. Sangaiah, "A review on object detection in unmanned aerial vehicle surveillance," *International Journal of Cognitive Computing in Engineering* **2**, 215–228 (2021).

2 S.-J. Hong, Y. Han, S.-Y. Kim, *et al.*, "Application of deep-learning methods to bird detection using unmanned aerial vehicle imagery," *Sensors* **19**(7), 1651 (2019).

3 L. Wittstruck, I. Kühling, D. Trautz, *et al.*, "Uav-based rgb imagery for hokkaido pumpkin (cucurbita max.) detection and yield estimation," *Sensors* **21**(1), 118 (2020).

4 M. Wu, W. Xie, X. Shi, *et al.*, "Real-time drone detection using deep learning approach," in *Machine Learning and Intelligent Communications: Third International Conference, MLICOM 2018, Hangzhou, China, July 6-8, 2018, Proceedings 3*, 22–32, Springer (2018).

5 Y. Cai, D. Du, L. Zhang, *et al.*, "Guided attention network for object detection and counting on drones," *arXiv preprint arXiv:1909.11307* (2019).

6 W. Boonpook, Y. Tan, Y. Ye, *et al.*, "A deep learning approach on building detection from unmanned aerial vehicle-based images in riverbank monitoring," *Sensors* **18**(11), 3921 (2018).

7 A. Ramachandran and A. K. Sangaiah, "A review on object detection in unmanned aerial vehicle surveillance," *International Journal of Cognitive Computing in Engineering* **2**, 215–228 (2021).

8 J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," *arXiv preprint arXiv:2304.00501* (2023).

9 Z. Wang, T. Schaul, M. Hessel, *et al.*, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, 1995–2003, PMLR (2016).

10 R. Opromolla, G. Inchingolo, and G. Fasano, "Airborne visual detection and tracking of cooperative uavs exploiting deep learning," *Sensors* **19**(19), 4332 (2019).

11 S. Kapania, D. S. Saini, S. Goyal, *et al.*, "Multi object tracking with uavs using deep sort and yolov3 retinanet detection framework," *Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems* (2020).

12 Z. Zhu, C. Hu, C. Zhu, *et al.*, "An improved dueling deep double-q network based on prioritized experience replay for path planning of unmanned surface vehicles," *Journal of Marine Science and Engineering* **9**(11), 1267 (2021).

13 L.-Y. Lo, C. H. Yiu, Y. Tang, *et al.*, "Dynamic object tracking on autonomous uav system for surveillance applications," *Sensors* **21**(23), 7888 (2021).

14 C. H. Singh, V. Mishra, K. Jain, *et al.*, "Frcnn-based reinforcement learning for real-time vehicle detection, tracking and geolocation from uas," *Drones* **6**(12), 406 (2022).

15 J.-H. Park, K. Farkhodov, S.-H. Lee, *et al.*, "Deep reinforcement learning-based dqn agent algorithm for visual object tracking in a virtual environmental simulation," *Applied Sciences* **12**(7), 3220 (2022).

16 Y. Li, H. Yuan, Y. Wang, *et al.*, "Ggt-yolo: a novel object detection algorithm for drone-based maritime cruising," *Drones* **6**(11), 335 (2022).

17 H. Nguyen, S. Thudumu, H. Du, *et al.*, "Uav dynamic object tracking with lightweight deep vision reinforcement learning," *Algorithms* **16**(5), 227 (2023).

18 S. Liu, X. Li, H. Lu, *et al.*, "Multi-object tracking meets moving uav," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8876–8885 (2022).

19 J. Lv, Q. Shen, M. Lv, *et al.*, "Deep learning-based semantic segmentation of remote sensing images: a review," *Sec. Environmental Informatics and Remote Sensing* **11** (2023).

20 G. Wang, J. Lin, Q. Zhai, *et al.*, "Mamask: Multi-feature aggregation instance segmentation with pyramid attention mechanism," *The Institution of Engineering and Technology* , 1341–1348 (2022).

21 Y. Pan, C.-A. Cheng, K. Saigol, *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *arXiv preprint arXiv:2002.00444* (2020).

22 W. N. . Y. C. Si, W., "Composite dynamic movement primitives based on neural networks for human–robot skill transfer," *Springer* , 23283–23293 (2021).

R Kiruthiga is a research scholar at National Institute of Technology, Tiruchirapalli. She received her B.E and M.E degrees in Computer Science and Engineering from Anna University in 2012 and 2016, respectively. Her current research interests includes computer networks, Machine Learning, and Deep Learning.

B Nithya received the Ph.D. degree from National Institute of Technology,Trichy, India in 2015. She is working as an Assistant Professor at the National Institute of Technology since 2007. Her research interest is in the Wireless Sensor Networks, the Internet of Things, Wireless Body area networks, Unmanned Aerial Vehicle, Internet of Vehicles, Advanced Data Structures, and Optimization Algorithms.

S Martin Prabhu is a Data science student in St Joseph's College, Trichy. His research interest is in the Machine learning Deep Learning, Unmanned aerial vehicle, Internet of Things, Advanced Data Analytics, Webscraping and Data Mining.

List of Figures

List of Tables