Drone collision avoidance in indoor environment

Mariia Makarova [1] **Ahmed Baza** [1] **Ekaterina Dorzhieva** [1] **Ayush Gupta** [1]

Abstract

The main goal of this study is to demonstrate the approach of achieving drone dynamic and static collision avoidance in an indoor environment using deep neural networks. The system will consist of a swarm of drones, a camera mounted above them, and static obstacles. Images from the camera are segmented into three groups: drones, obstacles, and floor. For the task of semantic segmentation, manual annotation is needed for our small custom dataset, which we use for training several neural networks. Computer vision algorithms process the segmented image and return the coordinates of the obstacle relative to the drone. Then we train the RL NN on the coordinates of the drones and static collisions and get the possible safe actions of the drones in real-time. We evaluate deep learning models trained on both synthetic and real data and present a new dataset that comprises both.

Github repo: https://github.com/makmary/Skoltech-ML-2022-Drone-Collision-Avoidance-In-Indoor-Environment
Video presentation: Video presentation

1. Introduction

Lately, Research involving drones applications is getting a lot of attention in the scientific community due to the various benefits drones can offer. Drones high mobility, mechanical simplicity, and low cost of production allowed them to offer solutions in a vast group of different fields. In remote sensing, one or more drones are used to explore and collect data about an area of interest providing researchers with data sets that weren't easily accessible before (Abualigah et al., 2021). In agriculture and climate changes drones were used for monitoring the health of planets and recently used for planting trees in remote areas that are not easily accessible by humans for the goal of reviving the forests (Stone, 2017). Drones applications are not just limited to the outdoor environments, indoor drones applications are increasingly varying. warehouses inventory management, intra-logistics, and warehouse Inspection are becoming a very dangerous and challenging task for human operators with the growth of the warehouses' sizes over the year to cope with the supply, drones offer a fast and reliable solution in this regard speeding the management process and replacing human operator in such hazardous environment(Wawrla et al., 2019). Other applications of drones include urban areas surveillance, maps building, structural inspecting and monitoring, photography, civil and personal use, arts and light shows, and defense applications.

Collision avoidance in an indoor environment plays a very important role in extending the flight time and payload carrying capabilities of drones. This field of study has brought many advancements in recent years. Drones can operate more autonomously by tackling problems such as obstacle avoidance and autonomous navigation. A major challenge that remains, however, is to ensure collision avoidance within the swarm itself.

To compensate for these collisions between the drones and the obstacles in the environments techniques like image segmentation and reinforcement learning are used.

Image segmentation is the process of subdividing a digital image into multiple image segments. They are also known as image regions or image objects (set of pixels). Semantic segmentation refers to the process of linking each pixel in an image to a class label. These labels could include a person, drone, box, and other objects.

In computer vision, most image segmentation models consist of an encoder-decoder network as compared to a single encoder network in classifiers. The encoder encodes a latent space representation of the input which the decoder decodes to form segment maps, or in other words maps outlining each object's location in the image.

We can think of semantic segmentation as image classification at a pixel level. For example, in an image that has many cars, segmentation will label all the objects as car objects. However, a separate class of models known as instance seg-

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Mariia Makarova <Mariia.Makarova@skoltech.ru>, Ahmed Baza <Ahmed.Baza@skoltech.ru>, Ekaterina Dorzhieva <Ekaterina.Dorzhieva@skoltech.ru>, Ayush Gupta <Ayush.Gupta@skoltech.ru>.

Final Projects of the Machine Learning 2022 Course, Skoltech, Moscow, Russian Federation, 2022.

mentation can label the separate instances where an object appears in an image. This kind of segmentation can be very useful in applications that are used to count the number of objects or find the best position to land.

2. Related Work

Obstacle avoidance is a fundamental part of the operation of an unmanned aerial vehicle. Currently, traditional algorithms for the safe movement of robots are widely used, such as artificial potential fields (Khatib, 1986). Obstacle avoidance path planning algorithm is undoubtedly one of the most important things to have a safe autonomous flight for a swarm of drones. This can be achieved by using the Rapidly-exploring Random Tree (LaValle & James J. Kuffner, 2001) planning algorithm which will be used for creating an obstacle-free path for the swarm of sUAS.

One of the traditional algorithms' problems is that it naturally doesn't guarantee optimality. Due to the limitations of a drone's flight time, an optimum path is much desirable for such applications. To achieve higher convergence rates, this problem can be solved by using informed RRT* (Gammell et al., 2014) which is becoming a hot topic in planning algorithms. Informed RRT* eliminates the unnecessary search by using a heuristic approach focusing on the area between current and target state. Besides that, most onboard computers have significantly limited calculating power and resources, which also constrain the possibility of successful maneuver to avoid an obstacle. In this case, classic collision avoidance algorithms, such as Artificial Potential Fields based on Machine Learning (ML) can offer a safer option (Haksar & Schwager, 2018).

Nevertheless, the considered algorithms often do not respond to fast-moving dynamic obstacles. Björn Lindqvist et al. (Lindqvist et al., 2020) proposed a projectile trajectory prediction method, which solves this problem without the use of Machine Learning tools, the maximum number of drones involved in the study is 2.

With an increase in the number of agents, in many situations, it is more appropriate to use the Reinforcement Learning method, since it allows you to teach agents to respond to any changes in the environment, numerous studies support its usefulness (Doukhi & Lee, 2021; Hsu & Gau, 2022; Qiao et al., 2008). An important task when working with a swarm in any practical problem is formation control (Dimarogonas & Kyriakopoulos, 2005), which is also achievable with RL-based algorithms (Zuo et al., 2010).

Deep Reinforcement Learning improves problem-solving using a powerful non-linear function approximation(Ouahouah et al., 2022), (Wang et al., 2020), for example, Sihem Ouahouah et al. used Deep Q-Network (Ouahouah et al., 2022). In this article, we will use the Actor-Critic algorithm (Mnih et al., 2016) for multi-agent environment (Lowe et al., 2017), as it allows you to combine Policy-Based and Value-Based RL methods, due to which it solves complex problems. Our Angry Birds-like RL shooting approach was reviewed by Ekaterina Nikonova Jakub Gemrot (Nikonova & Gemrot, 2019), however, they trained the agent to make the most successful shot that scores the most points, while we oppositely train the agents to avoid obstacles by dodging the shot.

3. Algorithms and Models

3.1. Image Segmentation

3.1.1. Dataset

Since for Image segmentation, a well-labeled dataset is required for the training of the model to achieve high accuracy on detections. Our custom dataset focuses on semantic understanding of the indoor environment for increasing the safety of autonomous drone flight and landing procedures. The imagery depicts several obstacles, red and blue boxes, a landing station for a drone, and drones themselves from nadir (bird's eye) view acquired at an altitude of 1 to 3 meters above the ground. A monocular camera was used to acquire images at a size of 1920x1080px, which were later converted to TIFF format. The training set contains 29 images and the test set is made up of 19 images. An example of such an image is shown in figure 1.

This data is considered to be very small, therefore each image has been divided into small images using python library `patchify` to produce more data to fit it to the models.

3.1.2. Data annotation

Apeer segmentation tool was used to annotate each image we used the annotation tool APPER. It is a cloud-based platform for microscopy image analysis and processing. It provided enough automated solutions to satisfy our needs.

3.1.3. Data augmentation

Various deep learning models often implement a data augmentation stage to reduce overfitting and improve performance in imbalanced class problems. The most common approach is to enlarge the dataset by using transformations such as divide, scale, rotate, and translate. The transformations we used are divide and rotate (images were taken by different angles, so we would get rotated images beforehand). Each image in the training and validation sets is divided into 16 non-overlapping sub-images and has identical sizes of 128x128 pixels.

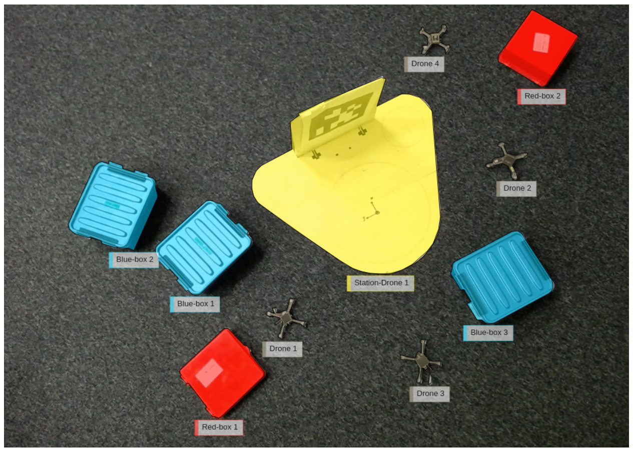

Figure 1. Annotating image in APEER tool

3.1.4. Convolutional neural network architecture

Convolution neural networks are used to solve classification problems, where we need to classify our images into one of many classes or perform image segmentation using architecture called U-net. We experiment with three different models and evaluate their performance, first two models with U-net architecture were trained from scratch, the second one was trained with ResNet34 as the backbone. This network and training strategy relies on the use of data augmentation to learn from the very few annotated images effectively. The U-net uses a usual contacting network by successive layers, where the pooling operations are replaced by the upsampling operators. Hence these layers increase the resolution of the output. A successive convolutional layer can then learn to assemble a precise output based on this information. Its architecture can be broadly thought of as an encoder network followed by a decoder network. Unlike classification where the result of the deep network is the only important thing, semantic segmentation not only requires discrimination at pixel level but also a mechanism to project the discriminative features learned at different stages of the encoder onto the pixel space.

CNN is rarely trained with weights that have been randomly initialized. Instead, it is preferable to use a convolutional neural network that has been trained previously on a very large dataset. ImageNet is a good example, as it contains more than 14 million images and more than 21,000 categories.

The pre-trained model we use is built from the ResNet34 model, where the number of network layers is 34, with weights pre-trained on ImageNet. We apply fine-tuning, truncating the top layer of the pre-trained network, and replace it with a new softmax layer, which in our case has 41 categories (species) and calculates the energy from the final property map along with the cross-entropy function and categorical focal loss in simple U-net architecture and categorical focal loss with dice loss of classes weights in U-net with ResNet34 as backbone. The convolutional neural network automatically learns by itself the features which are then classified through a softmax layer. We also experiment with others models such as VGG16 and VGG19, but the best results were obtained with ResNet34.

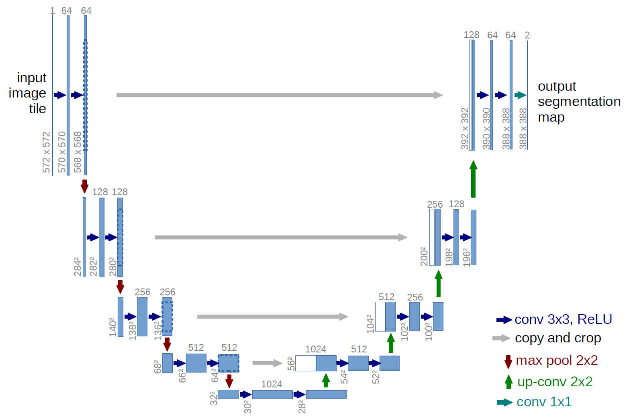

Figure 2. U-net architecture (an example image with a resolution of 32x32 pixels is the lowest). Each blue square corresponds to a multichannel property map. The number of channels is shown at the top of the square. The x-y dimension is given at the bottom left of the square. The white squares are copies of the property map. Arrows represent different operations.

The cross entropy loss at each point is give by:

$$L_{CE} = -\sum_{i=1}^{n}(t_i * log(p_i)) \tag{1}$$

where, n is the number of classes, t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

The categorical focal loss at each point is given by:

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma} log(p_t) \tag{2}$$

where, α_t is the weights of the classes and γ controls the shape of the curve, the higher the value of γ the lower the loss for a well-classified data.

The dice loss at each point is given by:

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \tag{3}$$

where, p_i and g_i represent pairs of corresponding pixel values of prediction and ground truth, respectively.

The implementation was done in Keras with Tensorlow as the back-end engine. The inputs for all models are 128×128 pixel with five channels for our 5 custom classes (0 - background, 1 - blue box, 2 - drone, 3 - station, 4 - red box) with information of their probabilities. When model predicts new mask, in algorithm it chooses the channel with the biggest probability for such multiclass task.

3.2. RL based Path Planning

We implemented the drone avoiding static obstacles and reaching the drones station using the Actor Critic algorithm.

3.2.1. Open AI Environment

An environment based on the Open AI Gym was created to train the agent. There is an agent in the environment at the same time - a drone, 4 obstacles and a drones station, whose positions along 2 axes are obtained from the previous segmentation step. We mean that the obstacles are of infinite height, the height of the drone and the target is known.

The coordinates of objects are the states (s_t) of the environment, at each moment the agent performs an action (a_t) that changes the state. Action refers to the speed of the drone - a vector in 3D space. After performing each action, the agent receives a reward(r_t) that penalizes him for the collision, passive actions and encourages for reaching the goal as in Eq. (4).

$$r_t = \begin{cases} -0.5 & \forall d_i < d_{collision} \\ 1 & \forall d_t < d_{target} \\ -2.5 * 10^{-4} & \end{cases}, \tag{4}$$

where $d_{collision}$, d_{target} are minimum distances of collision and reaching the target point. The agent's actions are calculated from the normal distribution as in Eq. (5) as it operates in a continuous action space.

$$a_t \sim \pi(a|\mu,\sigma) = N(a|\mu,\sigma), \tag{5}$$

where $\pi(a|\mu,\sigma)$ is the current policy, μ and σ are mean and variance of policy. Value of current state is calculating according Bellman equation (6).

$$V(s_t) = r_t + \gamma V_(s_{t+1}), \tag{6}$$

where γ is the discount factor, that measures importance of future rewards, $V_(s_{t+1}$ is the value of successor state.

3.2.2. Actor Critic Method

We applied Actor-Critic (A2C) approach due to its ability to learn simultaneously the policy and value functions. The architecture (Fig. 3 consists of 2 models, each of which takes the current state of the environment. The Critic model evaluates the value function of the state, the Actor model calculates the policy according to its state. We used various activation functions at the output of the model: hyperbolic tangent to calculate μ to get values in the range $[-1, 1]$, and the softplus for σ to get non-negative values .

The work of the Actor Critic (A2C) algorithm begins with replenishing the replay buffer (R) by performing random actions and the corresponding it feedback from the environment.

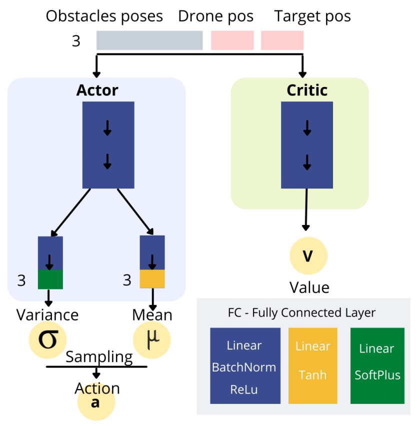

Figure 3. The Actor-Critic architecture.

The Critic loss function is calculated as a mean squared error of target and current value Eq. (4).

$$adv = V^*(s_t) - V_\pi, \tag{7}$$

$V^*(s_t)$ is the optimal value and V_π is the current value.

$$CriticLoss = -MSE(adv). \tag{8}$$

The Loss function is calculated as a log-probability Eq. (10).

$$Entropy_i = \sqrt{2\pi e\sigma_i^2}. \tag{9}$$

$$\log_{\pi\theta}(a|s) = \sum_{n=1}^{k}(-\frac{(a-\mu)^2}{2\sigma^2} - \log\sqrt{2\pi\sigma_i^2} + Entropy) \tag{10}$$

where k is the dimension of action space.

We add Entropy to Actor loss so that agents continue to explore the environment and do not stop at choosing the same actions Eq. (11).

$$EntropyLoss = \beta(-\frac{\sum\frac{\log(2\pi\sigma_i^2)+1}{2}}{batchsize}), \tag{11}$$

Algorithm 1 Actor-Critic Algorithm for drones path planning

```
1: Randomly initialize critic networks $V(s)$ and actor networks $\pi(s)$ with weights $U$ and $\theta$ for each agent. Initialize environment $E$ and replay buffer $R$.
2: for episode=1, M do
3:     for $t <= N$ do
4:         Sample action $a_t \sim \pi^\theta(a|\mu,\sigma) = N(a|\mu,\sigma)$ according to current policy.
5:         Execute action $a_t$ with $E$ and add current state $s_t$, reward $r_t$, action $a_t$, next state $s_{t+1}$ and $done \in True, False$ to replay buffer.
6:         if Train then
7:             Sample random batch from $R$.
8:             $\pi^\theta(s) \leftarrow$ actor network and value $V_\pi^U$ from critic network.
9:             Calculate $V_i^*(s_t)$ via Eq. (6).
10:            Calculate $Entropy$ via Eq. (9).
11:            Calculate $\log_{\pi\theta}(a|s)$ via Eq. (10).
12:            Calculate $EntropyLoss$ via Eq. (11).
13:            Update critic network by minimizing loss $CriticLoss$.
14:            Update actor network by minimizing loss $ActorLoss$.
15:        end if
16:        if done then
17:            break
18:        end if
19:    end for
20: end for
```

where β is the hyperparameter that controls the influence o entropy loss, $batchsize$ is the number of states for training

$$ActorLoss = \frac{\sum \log_{\pi\theta}(a|s)}{batchsize} - EntropyLoss. \quad (12$$

4. Experiments and Results

By following this link clear experimental setup with a fully reproducible code can be found for Image Segmentatior and RL based collision avoidance parts of this study.

4.1. Image segmentation

To test our prediction and compare it with known mask we performed calculations on an image taken from the test set The exemplar result of the prediction is shown in Fig.7. For a visual comparison we also provide the original image and its corresponding mask. Given imperfect segmentation, this example does show that the algorithm successfully detects drones, station, red and blue boxes. When there are few drones, boxes and/or station in an image and they are well separated from each other, the detector performs almost very well. In case of drones located on the station (a case of a drone intersecting a station), the model does not predict drone very well, further improvements are required, specifically in choosing model hyper-parameters and better image annotation, to achieve better performance by having such imbalanced data.

We performed several experiments with our three models, which are presented in Fig.4. We tried different values for hyper-parameters such as batch size and number of epochs. The best result we achieved was 98.49% for accuracy shown in Fig. 7. For it we use a learning rate of lr = 0.0001, with Adam optimizer and 50 epochs.

The evaluation metric chosen is Intersection-Over-Union (IoU), also known as the Jaccard index. Pixel accuracy gives the percent of pixels in the image that are classified correctly. Even though the metric is good for image segmentation it can become completely useless when the classes are imbalanced and in our case since in accordance to the drone

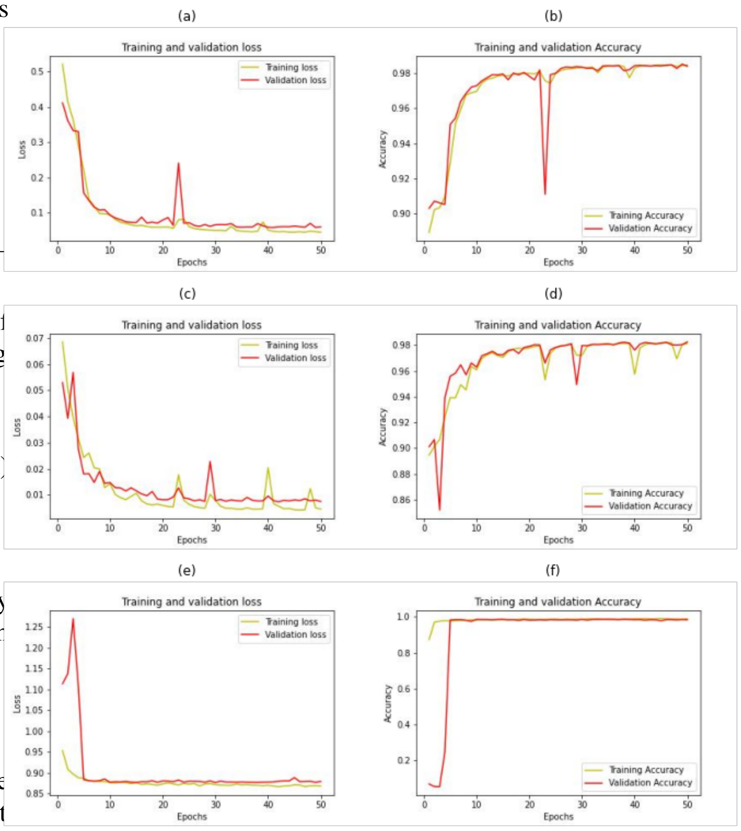

Figure 4. The figure shows training and validation accuracy and loss for different models: (a) loss for U-net, (b) accuracy for U-net, (c) loss for U-net with focal loss, (d) accuracy for U-net with focal loss, (e) loss for U-net with ResNet34 as a backbone, and (f) accuracy U-net with ResNet34 as a backbone

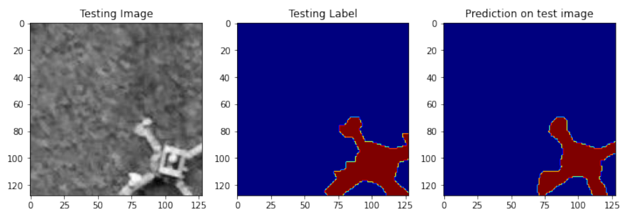

Figure 5. The prediction of our model on test data image. Here, the first picture correspond to original image, the second one to the training mask, the last one to the predicted mask. Red mask corresponds to one of the classes, in this example, it is a drone.

the background is very large which can thus decrease the quality of the results that are to be expected. To counter this class imbalance in the data, the IoU metric is considered. IoU is the intersection between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. For the case of multi-class classification, the mean IoU is calculated by taking the IoU of each class and averaging them. The quantitative comparison of our models' performance is presented in the Table 1. For segmentation task the best results is achieved by U-net with ResNet34 as a backbone providing mean IoU = 0.80. All three models did not predict red boxes and drones very well, the reason for that can be imbalanced data (smaller size and number of the drones and red boxes compared with other objects in the image) and imperfect data annotating and collecting (because of lighting and camera resolution), especially for drones on the drone station in Figure 7 (b).

After the semantic segmentation and detection of all the classes of the object present, the coordinates of each detected object are computed with computer vision techniques. For better contours of the detected classes morphological operations were used to remove the unnecessary noise, and also dilation was used to enhance the detected classes. In provided algorithm we used contour detection for different thresholds of the classes and to detect the centroid of the classes.

The computing infrastructure used is CPU: Intel i7 7th Gen 4.20GHz, GPU: Nvidia GTX 1080, RAM: 16GB and Storage: 256GB SSD.

4.2. RL based Path Planning

4.2.1. Collision avoidance

We start training process from implementing collision avoidance for drone with static fixed obstacles and target. The most important hyperparameters we tested in this experiment are β, a parameter that affects the agent's ability to explore the environment and not perform the same actions, and γ, a parameter measures the importance of future rewards.

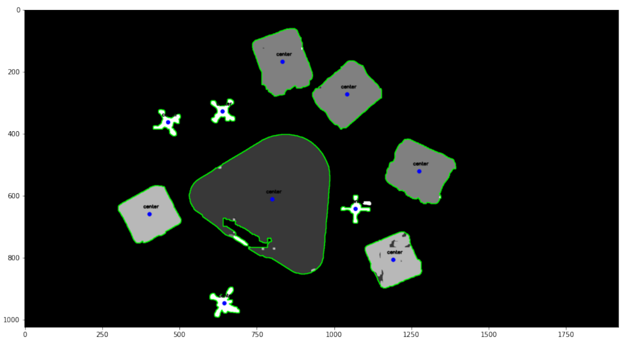

Figure 6. Segmented image with centroid coordinates that define a localization of the appropriate objects.

We trained the network for 50 epochs and tested collision avoidance by traversing the environment until the time runs out (2000 time steps) or until the drone hits an obstacle or boundaries. In this experiment, the drone was not rewarded for reaching the target. The training showed that the drone consistently performs moving without static predermined collisions after 35 epoch.

As can be seen from the Fig. 8, Critic loss decreases over time, which corresponds to an increase in value function and the cumulative reward, Actor loss fluctuates and decreases. We did not normalize the values of the models at the output, but did the normalization of the batches. Entropy loss is increased, which allows the agent to explore the environment.

4.2.2. Moving to target

The setup for experimentation is the same, now we're adding a goal reward function. After training on 100 epochs of experiments, 28 successful detections out of 100 of achieving the goal and avoiding obstacles were revealed. Experiments with the architecture of the neural network and changing the reward function slightly increase the result by 4% (32/100). The Fig/ 9 shows that the model stops training not reaching the award. The implementation of the algorithm still needs improvement and testing.

5. Conclusion

We present a path planning for drones based on deep learning-based semantic segmentation, detection algorithms, and reinforcement learning. This work compares U-Net network architecture with its improved modifications. For semantic segmentation, U-net model with categorical loss, with focal loss and ResNet34 backbone were chosen. The simple unit model gave a mean IoU of 0.794, U-net with focal loss gave a mean IoU of 0.725 and U-net with ResNet34

Table 1. Per class IoU and mIoU for comparing performance of three model with U-net architecture

Models	IoU (Drone)	IoU (Red Box)	IoU (Blue Box)	IoU (Station)	IoU (Background)	mIoU
U-net	0.651	0.590	0.881	0.858	0.990	0.794
U-net (focal loss)	0.423	0.555	0.837	0.821	0.988	0.725
U-net (ResNet34)	0.699	0.576	0.885	0.845	0.991	0.80

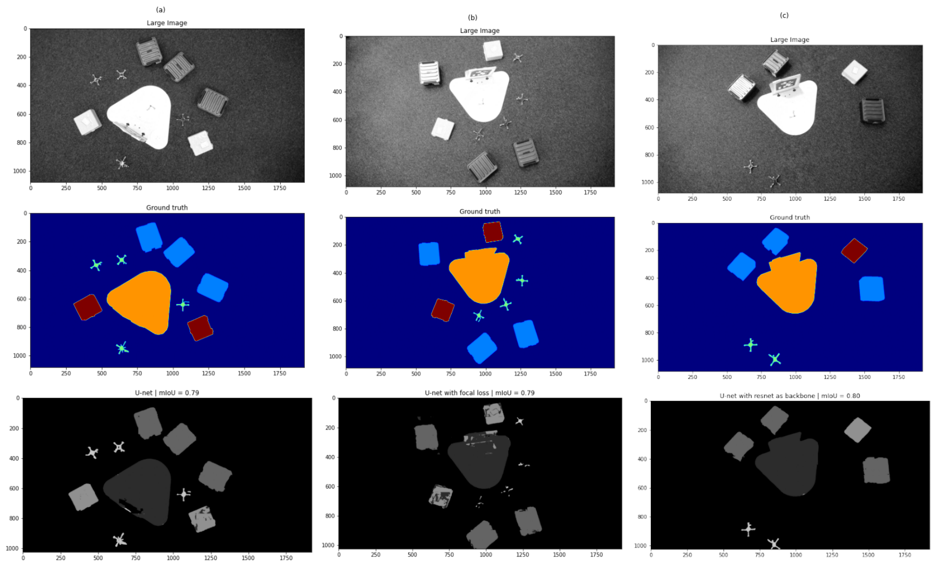

Figure 7. The figure shows the input image, ground truth labels and predictions on full image: (a) U-net, (b) U-net with focal loss, and (c) U-net with ResNet34 as a backbone

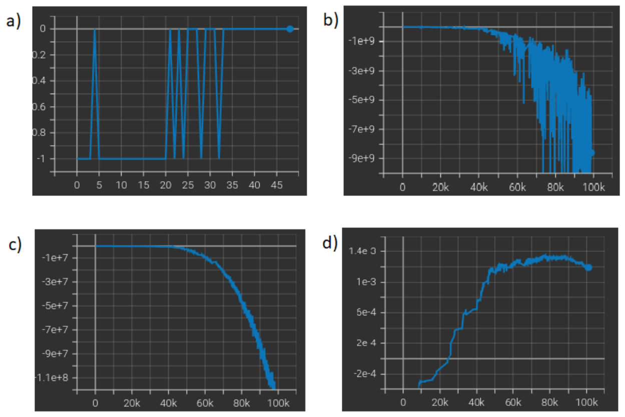

Figure 8. Collision avoidance training: a) Total score per epoch; b) Actor loss; c) Critic loss; d) Entropy loss.

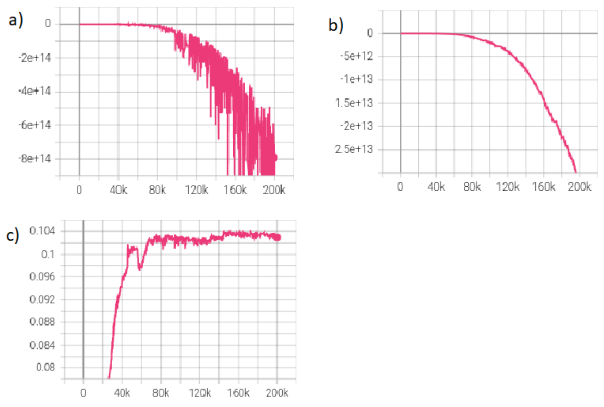

Figure 9. Moving to target training: a) Actor loss; b) Critic loss; c) Entropy loss.

gave an mean IoU of 0.80. The corresponding accuracy for these models were mostly near 98% though the metric IoU implied how the different classes are being balanced for the segmentation.

References

Abualigah, L., Diabat, A., Sumari, P., and Gandomi, A. H. Applications, deployments, and integration of internet of drones (iod): A review. *IEEE Sensors Journal*, 21(22):

25532–25546, 2021. doi: 10.1109/JSEN.2021.3114266.

Dimarogonas, D. and Kyriakopoulos, K. Formation control and collision avoidance for multi-agent systems and a connection between formation infeasibility and flocking behavior. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 84–89, 2005. doi: 10.1109/CDC.2005.1582135.

Doukhi, O. and Lee, D.-J. Deep reinforcement learning for end-to-end local motion planning of autonomous aerial robots in unknown outdoor environments: Real-time flight experiments. *Sensors*, 21(7), 2021. ISSN 1424-8220. doi: 10.3390/s21072534. URL `https://www.mdpi.com/1424-8220/21/7/2534`.

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic. *CoRR*, abs/1404.2334, 2014. URL `http://arxiv.org/abs/1404.2334`.

Haksar, R. N. and Schwager, M. Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1067–1074, 2018. doi: 10.1109/IROS.2018.8593539.

Hsu, Y.-H. and Gau, R.-H. Reinforcement learning-based collision avoidance and optimal trajectory planning in uav communication networks. *IEEE Transactions on Mobile Computing*, 21(1):306–320, 2022. doi: 10.1109/TMC.2020.3003639.

Khatib, O. The potential field approach and operational space formulation in robot control. In *Adaptive and learning systems*, pp. 367–377. Springer, 1986.

LaValle, S. M. and James J. Kuffner, J. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001. doi: 10.1177/02783640122067453. URL `https://doi.org/10.1177/02783640122067453`.

Lindqvist, B., Mansouri, S. S., Agha-mohammadi, A.-a., and Nikolakopoulos, G. Nonlinear mpc for collision avoidance and control of uavs with dynamic obstacles. *IEEE Robotics and Automation Letters*, 5(4):6001–6008, 2020. doi: 10.1109/LRA.2020.3010730.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1928–1937. JMLR.org, 2016.

Nikonova, E. and Gemrot, J. Deep q-network for angry birds. *CoRR*, abs/1910.01806, 2019. URL `http://arxiv.org/abs/1910.01806`.

Ouahouah, S., Bagaa, M., Prados-Garzon, J., and Taleb, T. Deep-reinforcement-learning-based collision avoidance in uav environment. *IEEE Internet of Things Journal*, 9(6):4015–4030, 2022. doi: 10.1109/JIOT.2021.3118949.

Qiao, J., Hou, Z., and Ruan, X. Application of reinforcement learning based on neural network to dynamic obstacle avoidance. In *2008 International Conference on Information and Automation*, pp. 784–788, 2008. doi: 10.1109/ICINFA.2008.4608104.

Stone, E. Drones spray tree seeds from the sky to fight deforestation. *National Geographic*, 2017.

Wang, D., Fan, T., Han, T., and Pan, J. A two-stage reinforcement learning approach for multi-uav collision avoidance under imperfect sensing. *IEEE Robotics and Automation Letters*, 5(2):3098–3105, 2020. doi: 10.1109/LRA.2020.2974648.

Wawrla, L., Maghazei, O., and Netland, T. Applications of drones in warehouse operations. *Whitepaper. ETH Zurich, D-MTEC*, 2019.

Zuo, G., Han, J., and Han, G. Multi-robot formation control using reinforcement learning method. In Tan, Y., Shi, Y., and Tan, K. C. (eds.), *Advances in Swarm Intelligence*, pp. 667–674, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-13495-1.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Mariia Makarova (25% of work)

- Reviewing literate on the topic (2 papers)
- Image annotation and augmentation
- Coding the main algorithm U-net
- Preparing the GitHub Repo
- Preparing of the report
- Preparing of the presentation

Ahmed Baza (25% of work)

- Reviewing literate on the topic (3 papers)
- Coding the main algorithm Environment & visualization
- Preparing the GitHub Repo
- Preparing of the report
- Preparing of the presentation

Ekaterina Dorzhieva (25% of work)

- Reviewing literate on the topic (3 papers)
- Coding the main algorithm Reinforment learning
- Preparing the GitHub Repo
- Preparing of the report
- Preparing of the presentation

Ayush Gupta (25% of work)

- Reviewing literate on the topic (2 papers)
- Image annotation and augmentation
- Coding the main algorithm U-net ResNet
- Computer vision for coordinates
- Preparing the GitHub Repo
- Preparing of the report
- Preparing of the presentation

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☐ Yes.
   ☑ No.
   ☐ Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: None

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☐ Yes.
   ☑ No.
   ☐ Not applicable.

   Students' comment: None.

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: It was done for image segmentation part of the work.

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: It was done for RL based collision avoidance part of the work.

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    ☐ Yes.
    ☑ No.
    ☐ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

    - ✓ Yes.
    - ☐ No.
    - ☐ Not applicable.

    Students' comment: None