A Lightweight Real-Time Person Detection System for Aerial Search and Rescue

Student Name: Alexander Read
Supervisor Name: Dr. Yona Falinie Abd. Gaus
Submitted as part of the degree of BSc Computer Science to the
Board of Examiners in the Department of Computer Sciences, Durham University

Abstract—The increasing use of camera drones by search and rescue (SAR) teams globally provides a significant opportunity for real-time person detection. However, state-of-the-art real-time detection methods have primarily been investigated for deployment on drones or devices with significant GPU resources. These hardware requirements are costly and impractical for SAR teams who only have access to older drone technology. We propose a real-time person detection system for SAR operations, and a Convolutional Neural Network (CNN) model which can operate on portable devices such as a laptop. The system works by receiving live video data from a drone and running our light-weight CNN model to perform real-time analysis. In the development of the system we make three contributions 1) We review existing SAR datasets and annotate biases in the available data 2) We introduce a synthetic dataset 'GTASar', for desert and alpine SAR 3) We deploy a trained YOLOv8-nano model with our proposed system and evaluate the performance of slicing adaptive inference as an enhancement. The final model demonstrated an impressive 83% on AP@0.5 across our test set and operated at 21 FPS on a MacBook Pro 2020. Compatible with any drone that can stream data, the system was validated in a SAR simulation where it successfully helped in identifying a 'lost' person.

Index Terms—Artificial Intelligence, Image Processing and Computer Vision, Search and Rescue, Real-time Systems

1 Introduction

Search and rescue (SAR) operations aim to locate and assist people who may be lost and in distress, and often in imminent danger. Historically, SAR missions have relied on helicopters to pinpoint individuals, which is a costly exercise. Therefore, the emergence of cost-effective unmanned aerial vehicles (UAVs) has been welcomed, in particular drones, leading to their growing popularity among SAR teams [1].

Different camera-equipped drones are available based on the location, budget and SAR teams' flying experience [2]. Certain drones contain pre-installed person detection algorithms that provide real-time detection, but SAR teams often do not use this technology as it can lag behind state-of-the-art and is unsuitable for different remote environments [3], [4]. Currently, research prioritises deploying larger person detection models either directly to drones or to ground systems with high-performance GPUs [5], [6], [7], [8]. Deploying a model directly to a drone means the drone needs to possess the hardware required to run the models, which is not appropriate for SAR teams who use older drone technology, particularly in Global South countries [9], [10], [11]. This method also reduces the battery time and, thus, the flight time of a drone, and in many countries, SAR teams have to seek permission to deploy artificial intelligence (AI) models directly to a UAV [12].

In this study, we present an AI framework that can be run on modern laptops or portable devices without the need for specific hardware or a powerful computer. In our proposed system, video data is streamed from a drone to the device running our proposed model. This means that 1) A device with substantial GPU resources is not required and 2) Our system works with any drone that is capable of streaming video data.

Our aim is to develop a system for remote SAR targeting three categories, each encompassing different terrains: Alpine SAR (Al-SAR) [13], for mountainous regions with hazards such as snow, ice, rocks, and steep slopes; Wilderness SAR (WiSAR) [14], covering undeveloped areas like forests, deserts, or grasslands; and Maritime SAR (MaSAR) [15], focusing on open waters (oceans, seas, large lakes) and coastlines. We refer to these terrain group definitions throughout this paper.

1.1 Computer Vision

Developing a SAR person detection system involves addressing a computer vision challenge, particularly person detection. Traditional algorithms for person detection trained Support Vector Machines (SVMs) to recognise the Histogram of Oriented Gradients (HOG) or scale-invariant feature transform (SIFT) [16] features of a person using large amounts of training data [17] [18].

However these methods struggled with intricate backgrounds, lighting variations, and object deformations, common in search and rescue (SAR) environments [19]. Additionally, these methods rely on handcrafted feature extraction algorithms, requiring manual tuning for specific environments or terrains.

Rapid developments have been made in using convolutional neural networks (CNNs) as an improved method for person detection [20]. CNN object detection algorithms are split into two groups: one-stage algorithms and two-stage algorithms. One-stage algorithms apply a single neural network to the full image. This network divides the image into

regions and predicts bounding boxes and class probabilities for these regions all in one pass. Two-stage object detection algorithms generate region proposals using methods such as selective search or a region proposal network. A separate classifier model then evaluates these selected regions to identify and locate objects.

Both two-stage algorithms, such as the Faster Regions with Convolutional Neural Networks algorithm (Faster-RCNN) [21], and one-stage algorithms, such as the Real-Time Object Detector algorithm (RTMDet) [22] and the You Only Look Once (YOLO) algorithm [23], have been successfully used to train person detection models [24], [25], [26], [27]. While two-stage algorithms offer better accuracy than one-stage algorithms, one-stage algorithms are less computationally expensive and have faster inference speeds [28].

Therefore, in this project, we trained and compared the YOLOv8 [29] (the latest version of YOLO), RTMDet and Faster R-CNN algorithms to develop the best-performing model for our proposed system. We integrate the smallest variations of the YOLOv8 and RTMDet architectures, dubbed YOLOv8-nano and RTMDet-tiny. These variations can run on devices using a CPU instead of a GPU, making them suitable for deployment to laptops and other portable devices. We also examined the performance benefits of slicing adaptive inference (SAHI) [30]. SAHI increases a model's ability to detect smaller objects by dividing an image into slices, performing inference on each slice, and then merging the detections.

1.2 Challenges

Training a model to detect individuals across diverse terrains presents significant challenges due to the limited availability of SAR data, particularly in mountainous regions and deserts [4]. Moreover, for a SAR person detection model to be effective, it must be trained on a wide range of data that accurately represents remote SAR operations. It should also be robust enough to handle harsh weather conditions, varying lighting, and lower image quality during real-time deployment [31].

1.3 Project Objectives and Deliverables

To develop our proposed system, we integrated five key components into this study. Initially, we evaluated the available SAR datasets [32], [33], [34], [35]. Subsequently, we compiled a synthetic dataset to address gaps in terrain groups. Following this, we trained three object detection algorithms—RTMDet-tiny [22], YOLOv8-nano [36], and Faster-RCNN [37]—on the comprehensive remote SAR dataset. We selected the best-performing model and assessed the performance enhancement provided by SAHI [30]. Finally, we established our proposed deployment system and tested the model on a MacBook Pro 2020 during a simulated SAR operation.

To evaluate the project we split it into a number of deliverables:

1.3.1 Basic Deliverables

- Analyse available SAR datasets and determine under-represented terrain groups.
- Construct an extensive dataset that can be used to train a machine learning model for SAR person detection.
- Train a person detection model by employing state-of-the-art algorithms
- Evaluate and compare the performance of the models when executed on a laptop with limited GPU capabilities.

1.3.2 Intermediate Deliverables

- Select the best performing model for real-time deployment.
- Investigate optimisations to the model (SAHI) on top of the best performing model.
- Propose a system of deployment for SAR operations.

1.3.3 Advanced Deliverables

- Deploy the system in a simulated search and rescue operation.
- Evaluate the usefulness of the tool in the simulation.

2 Related Work

In this section, we provide a brief overview of the most commonly used object detection algorithms, their application in detecting individuals for search and rescue (SAR) operations, and the limitations encountered within the systems where they have been implemented.

2.0.1 You Only Look Once Detector (YOLO)

The YOLO (You Only Look Once) series consists of single-stage detection algorithms that perform object detection in a single network pass, as shown in Figure 1.

The first iteration, YOLOv1, introduced a novel approach to object detection and works by dividing an image into a grid of cells, where each cell is responsible for predicting object's whose centres fall within it. Then, within each cell, the YOLO algorithm predicts multiple bounding boxes, each with a confidence score which indicates the likelihood of an objects presence. To refine these predictions, YOLO applies non-max suppression [38]. Different computer vision algorithms have different 'backbones', the architecture of the convolutional neural network used for feature extraction. A 'feature map' is a term used to describe the features extracted by convolutional layers; feature maps extracted by a set of layers feed into subsequent layers in the network. The initial version of the YOLO algorithm used a lightweight CNN with 24 convolutional layers. This architecture was simple and fast but less accurate than other algorithms like Faster RCNN [21].

YOLOv2 [39] introduced an improved backbone 'Darknet-19' and was utilised in work to train a model for SAR. However in this work the model faced challenges in identifying multiple people, detecting individuals at a distance, and needed a powerful GPU for real-time inference [40], [23].

The deployment of a YOLOv3 [42] model for person detection directly to a drone was later explored for SAR [43],

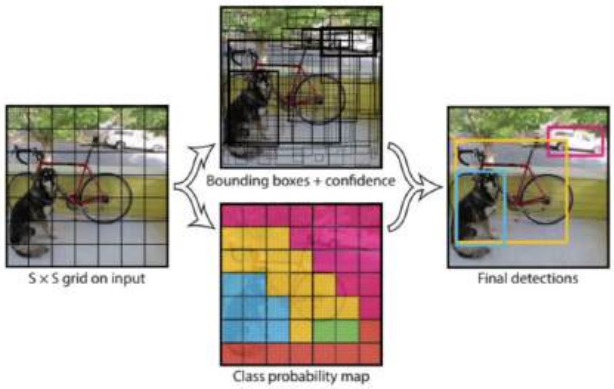

Fig. 1: Diagram showing the YOLO architecture [41]

[6]. Although effective, the requirement for an embedded GPU like the 'Nvidia Jetson TX1' [44] or the 'Jetson Xavier NX' [45] to run inference directly on a drone meant that using such systems required specific hardware. Tiny YOLOv3 [46], a smaller variant of YOLOv3, was designed for deployment on devices with lower-powered or less expensive GPUs. It used fewer layers, and made use of depth-wise separable convolutions [47], a type of convolution operation that has become popular for smaller architectures [48]. This algorithm was used to train a SAR model to run directly on an operator's mobile phone in a system trialled by Police Scotland [24]. However this only worked with the DJI SDK, and could only operate at 6.8 FPS (frames per second), not real-time speeds.

The YOLOv5 algorithm used in conjunction with slicing adaptive inference (SAHI) [30], was used to detect a range of small objects, including humans, in large images. This example is notable as the addition of SAHI showed an improved mean average precision (mAP) score of over 5% on the study's test data.

Subsequent versions of YOLO have been developed and each version has introduced minor improvements. The most recent iteration, YOLOv8 [29], has the best performance [49]. The main change to YOLOv8 compared to its predecessors is that it uses a complete anchor-free detection system [50]. Anchor boxes are predefined bounding boxes used to detect objects at different scales and aspect ratios in an image and were used in all subsequent versions of YOLO. In YOLOv8 the model learns to predict the width and height of detected objects without using any anchor box weights. Varied drone camera angles require recognising people in multiple poses and perspectives where anchor box weights are hard to learn, as such the use of YOLOv8 for small person detection has shown to be superior to its main predecessor YOLOv4 [51].

YOLOv8 has been used to train a model to detect people in thermal imagery for SAR [27]; however, despite its capabilities, YOLOv8 has yet to be deployed directly on ground devices for SAR, especially on devices with limited GPU resources. Therefore, in this paper, we seek to explore the capability of YOLOv8 on edge devices for SAR.

2.0.2 Single Shot MultiBox Detector (SSD)

The Single Shot MultiBox Detector (SSD) [52] algorithm works similarly to the YOLO algorithm, however SSD does not use a fixed grid to detect objects; instead, it places predefined anchor boxes of various sizes and aspect ratios at each location on multiple feature maps. SSD models have been proposed for SAR person detection [26], [5]. However, once again, this work only deployed the SSD models to hardware, with substantial GPU resources. Additionally the latest versions of YOLO have outperformed SSD in real-time detection [53], thus we chose not to include this algorithm in our experiments.

2.0.3 Real-Time Detector (RTMDet)

Another single stage algorithm, RTMDet (Real-Time Detector) [22], also works similarly to the latest versions of YOLO. However it differs in its approach to feature processing. Unlike YOLO, which uses a uniform feature extractor across all scales, RTMDet uses modules that handle various object sizes and scenes, providing an edge in performance where YOLO might struggle with granularity and occlusion. The algorithm has beaten the performance of YOLOv8 in small person detection tasks [54], and offers the benefit of being free from licensing fees, unlike YOLOv8. In our experiments, we implement the RTMDet algorithm to determine the best performing SAR person detection model.

2.0.4 Faster Region-based Convolutional Network Detector (Faster R-CNN)

The Faster Region-based Convolutional Network (Faster R-CNN) [21] is the most used object detection algorithm with a two-stage architecture. It builds upon the Fast R-CNN [37] algorithm, which itself expanded upon the original R-CNN paper [55]. As previously discussed, Faster R-CNN works by running a region proposal network (RPN) that scans the input data to identify regions of interest. These identified regions are then processed by a classifier network to accurately predict if an object is in fact present in that region of the image, this process is shown in Figure 2. Faster R-CNN has been used to train person detection models which have been deployed in SAR scenarios with successful results [25], [5], [35] however in some cases the models developed were not proposed for real-time use, whilst in other cases the proposed systems required extensive GPU resources for real-time inference. In our experiments, we implement the Faster R-CNN algorithm to determine the best performing SAR person detection model.

2.1 Datasets

2.1.1 Person Detection Datasets for Search and Rescue

Training object detection algorithms effectively require data that is relevant to the intended application. Despite the growing use of drones, acquiring suitable SAR data for remote and often visually obscured environments—where such operations are typically conducted—remains a significant challenge. Among the few notable SAR datasets are the HERIDAL Dataset [35], the WiSAR Dataset [34], and the SeaDroneSee Dataset [33]. Additionally, we identified the SARD Dataset [32], which, although smaller, consists of images from a single simulated SAR operation.

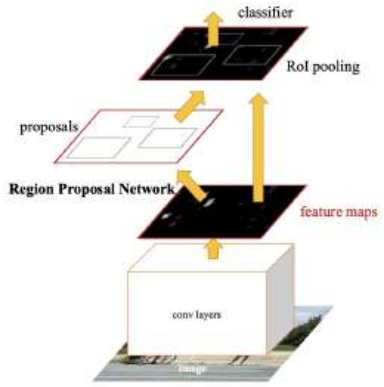

Fig. 2: Diagram showing the Faster R-CNN architecture [21]

This has meant that many SAR models have been trained on the same datasets [56], and thus will fall short to the same biases present in these datasets.

To enrich limited availability of real data, synthetic data—data created using computer generation—has been employed to train object detection models. This approach reduces the reliance on real-world data. Synthetic data has proven effective in applications such as crowd counting and pedestrian tracking [57], [58]. Frequently, video games or game engines are utilised to generate lifelike data. A notable instance is the use of the video game *GTA 5* [59], which has been leveraged to create synthetic datasets for training models in object recognition and scene reconstruction [60].

The use of synthetic training data in the field of CNNs for various applications has been explored extensively [61], [62], [63] yet its application in human detection for SAR remains largely unexplored.

2.2 Summary of Related Work

Limited research has been dedicated to deploying search and rescue (SAR) person detection models on devices with limited GPU resources, regardless of the object detection algorithm employed. Previous attempts to deploy on portable devices have yielded only modest performance, such as operating at 6.8 FPS, or have been limited to specific drone models, like DJI drones. This project distinguishes itself by focusing on three key objectives: 1) Training a state-of-the-art person detection model that operates in real-time on modern laptops and devices; 2) Incorporating synthetic data into the training process; and 3) Deploying the SAR person detection model on a system compatible with any drone.

3 Methodology

In this section we propose our framework to deploy a SAR person detection model on devices with limited GPU resources. Section 3.1 details the initial collection, annotation and evaluation of available SAR datasets. Section 3.2 details the collection and annotation of our novel synthetic dataset 'GTASar', and evaluation of the final dataset used. Section 3.3 details the data augmentation applied to the final dataset. Section 3.4 describes the object detection algorithms selected and the hardware used to the train the models. Section 3.5 describes our proposed deployment system.

TABLE 1: SAR Labels used to tag images in dataset

Label	Description
Alpine SAR (AlSAR)	mountainous - snow, ice rocks
Wilderness SAR (WiSAR)	forests, deserts, or grasslands
Maritime SAR (MaSAR)	open waters and coastlines.

TABLE 2: Sub-labels used to tag Wilderness SAR marked images

Label	Description
WiSAR-forest	forest environments
WiSAR-desert	desert environments
WiSAR-grassland	grassland environments

Section 3.6 describes how the simulated SAR operation was conducted. Finally, section 3.7 outlines the methods used to evaluate the project.

3.1 Initial Data Collection and Annotation

The initial step of our project involved gathering SAR images from four public datasets to create a large dataset for training a SAR model. As we processed the images from these datasets, we excluded those that were not relevant for training a person detection model. To identify any biases, we annotated all images with the remote SAR labels as shown in Table 1. Additionally, we assigned further sub-labels specifically to images pertaining to wilderness search and rescue, as shown in Table 2. This detailed labelling was particularly focused on wilderness search and rescue images to highlight the scarcity of data from desert environments, justifying the need for the synthetic dataset introduced later in our study.

3.1.1 SARD Dataset [32]

The SARD dataset contains 2086 annotated images (See Appendix A.4). It includes images containing people on macadam roads, quarries, high grass and forest shade, all captured within the same geographic region. It is labelled with 266 AlSAR and 2084 WiSAR labels—all WiSAR labels are sub-labelled as grassland.

3.1.2 SeaDroneSee Dataset [33]

The SeaDroneSee dataset contains over 10,000 annotated images (See Appendix A.5) from MaSAR scenarios. These images contain both people and boats, and some images from the original dataset exclusively containing boats have been removed; we use 5083 images from the dataset—all images in this dataset were given the MaSAR label.

3.1.3 WiSAR Dataset [34]

The WiSAR Dataset (WiSARD) comprises over 10,000 labelled images (See Appendix A.6). We use a subset of this dataset, removing thermal imagery. Some images not taken from a UAV were discarded. After these adjustments, we

Fig. 3: Images showing unannotated synthetic dataset

used just 2999 images from the dataset. The images used were given a total of 1983 AlSAR and 2012 WiSAR labels - 52% WiSAR labelled images were forest sub-labelled, with the remaining 48% labelled as grassland.

3.1.4 HERIDAL Dataset [35]

The HERIDAL Dataset contains 1577 labelled images from a wide range of environments (See Appendix A.7). The images contained 2 MaSAR labels, 919 AlSAR labels, and 1959 WiSAR labels, of which 48.4% were grassland sub-labelled, and the remaining 51.6% forest sub-labelled.

3.1.5 Dataset Analysis

Figure 4 shows the gathered composite dataset contains 5085 MaSAR labels, 3155 AlSAR labels, and 5950 WiSAR labels. Figure 5 shows that 49% of WiSAR labels were sub-labelled as grassland and the other 51% labelled as forest. This distribution highlights the under-representation of Alpine search and rescue (SAR) data. Furthermore, there is an absence of desert SAR data. To address these two issues, we have generated a synthetic dataset tailored to these environments.

3.2 Synthetic Dataset Collection and Annotation

We developed a synthetic dataset, termed 'GTASar,' utilising the video game *GTA 5*. Initially, we recorded short ten-minute gameplay videos capturing a player navigating on foot while another player piloted a helicopter above, simulating aerial surveillance. Frames were extracted from these video clips and annotated using the tool 'Roboflow' [64]. Subsequently, the extracted images were labeled according to the categories outlined in Tables 1 and 2. The dataset included images of three distinct characters, each depicted in three unique outfits. These characters varied across gender and races, to introduce diversity in the synthetic dataset.

Figure 3 shows images from the GTASar dataset. The final dataset is comprised of 1,469 images and contains 66 MaSAR labels, 499 AlSAR labels and 991 WiSAR labels. 90.2% of WiSAR labelled images are sub-labelled as desert terrain, 0.6% forest and 9.2% grassland.

Incorporating the synthetic dataset into the existing composite set resulted in a final dataset comprising of 13,214 images. While imperfect, the distribution of labels in the final dataset, as shown in Figure 4 and Figure 5, now more accurately reflects the desert and alpine terrains typically involved in remote SAR operations.

The dataset was divided into training, validation, and testing subsets at ratios of 70%, 15%, and 15%, respectively.

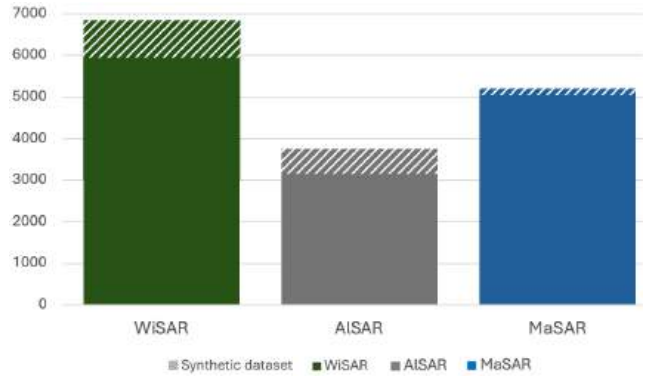

Fig. 4: Distribution of SAR Labels in the original and synthetic-enhanced dataset

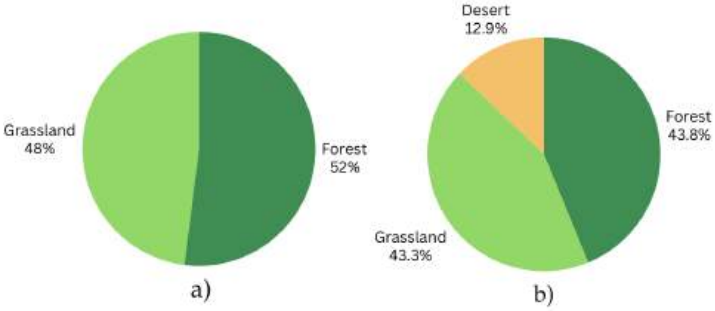

Fig. 5: a) Distribution of WiSAR sub-labels in original dataset b) Distribution of WiSAR sub-labels in final dataset with added synthetic data

We maintain the data division if the dataset comes with its own training, validation and testing sets. It is important to note that synthetic data was exclusively used in the training split, as it does not mimic real-life scenarios accurately enough for validation or testing purposes.

To further analyse the dataset, Figure 6 presents an annotation heat map which was used to identify any potential biases in the placement of annotations. To address the bias in central annotations we employed mosaicing techniques as well as other data augmentation methods prior to training the model. This ensured a more robust training process for the SAR person detection model.

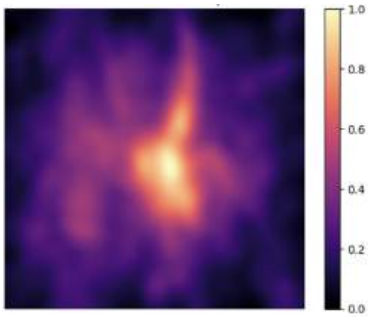

Fig. 6: Annotation heat map of final composite dataset

TABLE 3: Data augmentation applied to original dataset, (t)-exclusive to training set

Augmentation	Augmentation rate
Resize 640x640	100%
Pixelation and Compression	10%
90° Rotate: Clockwise, Counter-Clockwise (t)	1:3
Blur (2.5px) (t)	1:3
Mosaicing (t)	1:3

TABLE 4: Training Environment Configuration

Category	Configuration
CPU	Intel® Xeon® Processor E5-2623 v4
GPU	Tesla V100 (16 GB GPU memory)
System Environment	Ubuntu 20.04 Server
Frameworks	Ultralytics, MMDetection, MMYOLO
Programming voice	Python 3.8
Logging Framework	Weights and Biases

3.3 Data Augmentation

Several data augmentations were applied to the dataset to improve the model's ability to recognise patterns or features under different conditions. Streaming video data can present unique data augmentation challenges, and we tried to replicate these in our dataset; Figure 7 shows an example of pixelation and compression augmentation applied to replicate buffering artifacts. We also applied standard augmentations to the training dataset, shown in Table 3. An augmentation rate of 1:3 means that for each original image, three new versions were created through augmentation, with each type of augmentation having an equal likelihood of application.

Fig. 7: Example of original image (left) with compression augmentation (right)

3.4 Model Selection and Training Equipment

With many state-of-the-art object detection algorithms available, our strategy was to train three models using algorithms that optimise the balance between precision and inference speed. This selection process allowed us to identify the most efficient model for our SAR person detection tasks, particularly those that perform well on our composite dataset and experimental hardware, while meeting our near real-time inference requirements. Constraints included algorithm suitability for detecting smaller objects and overall computational efficiency.

For this project, we selected the 'nano' variant of YOLOv8, due to its efficiency and speed. We also implemented RTM-Det-tiny within the MMYOLO [22] framework and Faster R-CNN with a ResNet-50 backbone (Faster R-CNN r50 FPN) from the MMDetection [37] framework. Each of these models was chosen for its specific advantages in handling the demands of SAR person detection in varied and challenging environments.

To optimise training outcomes, we fine-tuned the hyperparameters for each selected algorithm, conducting training over 100 epochs with a batch size of 16. We systematically compared the models using the evaluation metrics described in Section 3.7. Throughout the training period, each model was evaluated on the validation set after every epoch to monitor progress and adjustments. Detailed information on the training process and the specific hyperparameters employed are shown in Table 7.

The training was executed on a machine provided by Paperspace [65], as outlined in Table 4. We utilised the 'Weights and Biases' tools [66] to track training progress and systematically record the metrics critical for thorough evaluation and analysis.

3.5 Model Deployment and Optimisation

Table 5 details the specifications of the 2020 MacBook Pro where each model was evaluated. As discussed in Section 3.7.1, to accurately assess the inference speeds of each model, FPS was measured by exporting all models to this laptop in a standardised format. While this study focuses on the MacBook Pro, the principles applied can be adapted to suit the hardware capabilities of any device used by a SAR team. Models were converted to the ONNX format, a platform-independent model format, and executed within the ONNX runtime [67] installed on the MacBook.

Additionally, the impact of using Slicing Adaptive Inference (SAHI) [30] in the deployment process was considered. This method involves dividing a larger image into smaller slices, performing inference on each slice, and then merging the detections. This technique has been demonstrated to enhance the detection of smaller objects [68], though it introduces an additional step in the inference pipeline. To determine the optimal baseline model, we applied SAHI to only the top-performing model, as it could slow down the inference speed and possibly affect real-time performance on specific devices. This was considered an optional enhancement.

The deployment system, as depicted in Figure 8, is designed to run inference on a live-streamed video feed. Most camera-equipped drones support streaming via an RTMP (Real-Time Messaging Protocol) Server [69], [70]. We propose processing frames from this stream in near real-time using the trained model on a SAR team's device. This setup enables SAR operators to monitor a drone's live

TABLE 5: MacBook Pro 2020 M1 Hardware Specifications

Component	Specification
Processor	Apple M1 (8-core)
Graphics	Integrated 8-core GPU
RAM	16 GB Unified Memory
Storage	1 TB SSD
Operating System	macOS Sonoma

TABLE 6: DJI Mavic Pro Specifications

Component	Specification
Drone Name	DJI Mavic Pro
Max Flight Time	27 minutes
Camera Resolution	12 MP
Video Resolution	4K @ 30fps
Control Range	7 km

feed with overlaid real-time inference, assisting in the quick identification of lost individuals.

3.6 Search and Rescue Simulation

A search and rescue (SAR) simulation was conducted to assess the system's effectiveness, as illustrated in Figure 9, depicting the 300m x 400m wilderness search area used for the simulation. The simulation employed a MacBook Pro as outlined in Table 5 and a DJI Mavic Pro 2020 drone, detailed in Table 6. The scenario involved one participant simulating a missing person by adopting various concealed positions while another participant, acting as the SAR operator, piloted the drone to search the designated area. The detection system was operated via the laptop to locate the 'lost' individual, whose position was unknown to the operator at the start of each simulation run. This exercise aimed to validate the system's functionality rather than perform an extensive quantitative analysis.

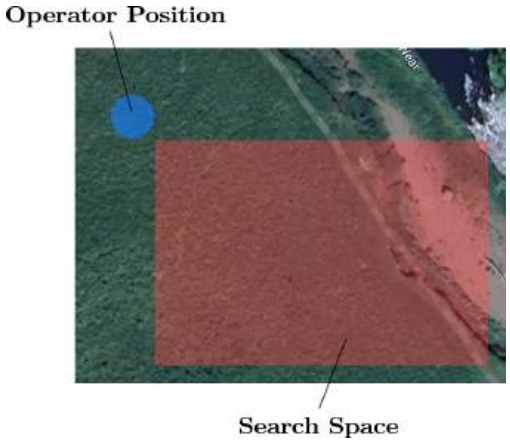

Fig. 9: Diagram showing wilderness SAR simulation setup

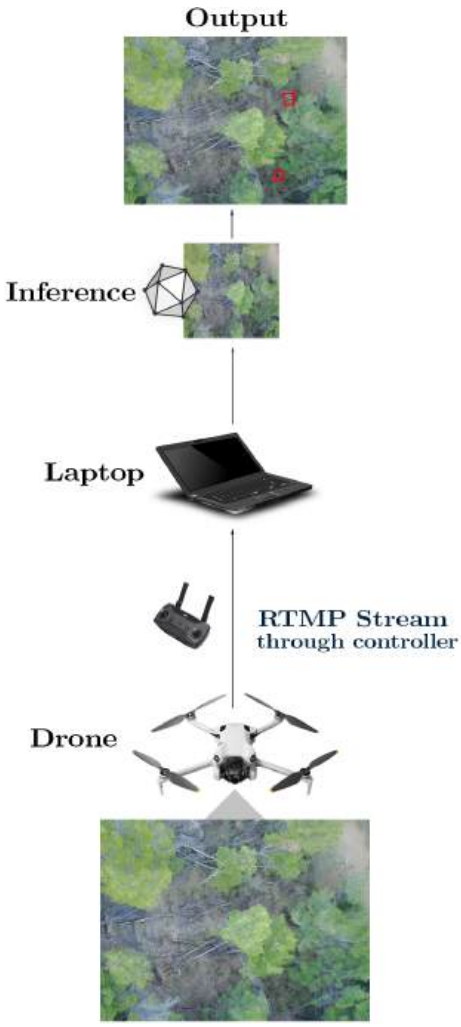

Fig. 8: Diagram showing proposed SAR detection system

3.7 Methods of Evaluation

3.7.1 Model Comparison Metrics

We evaluated each model on the test split of the dataset and measured average precision (AP) and average recall (AR) over different Intersection over Unions (IoU). We also measured two performance indicators, the frame rate (FPS) during inference and the network parameter volume (model size). A higher FPS suggests better real-time performance, while decreased model size indicates lower memory usage. Finally we compared the input size (pixels) that each model uses, as scaled up input images enable a network to observe more details within an image.

Precision is defined as the ratio of true positive predictions (TP) to the total number of positive predictions made by the model:

$$P = \frac{TP}{TP + FP}$$

where FP represents false positives.

Recall measures the proportion of actual positives correctly identified by the model, calculated as:

$$R = \frac{TP}{TP + FN}$$

with FN denoting false negatives.

Average precision AP is therefore calculated by averaging the precision scores that the model achieves on each image in the test set.

$$\text{AP} = \frac{1}{N}\sum_{i=1}^{n} P(i)$$

where N is the number of images

AP at different Intersection over Union (IoU) thresholds, defines what is allowed as a 'positive prediction'.

For example $AP@0.5$ measures the average precision where a prediction has to overlap the ground truth of the test image by 50%. Figure 10 shows an example of an accepted and unaccepted prediction with a 0.5 IoU threshold. We measure $AP@0.5$ as a percentage:

$$AP@0.5(\%) = AP(i = 0.5) \times 100$$

where AP(i) is the average precision at a single IoU threshold 'i'.

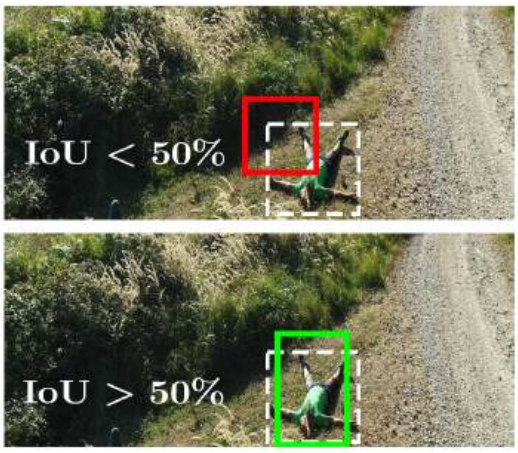

Fig. 10: Visual representation of IoU criteria

$AP@[0.5 : 0.95]$ measures the precision across a range of IoU thresholds, from 0.5 to 0.95 in increments of 0.05. For each threshold, the AP is calculated, and the results are averaged to provide a single metric:

$$AP@[0.5 : 0.95](\%) = \left(\frac{1}{10}\sum_{i=0.5}^{0.95} AP(i)\right) \times 100$$

where AP(i) is the average precision at a single IoU threshold 'i'.

AR is calculated by just averaging the recall achieved on each image in the test set:

$$\text{AR} = \frac{1}{N}\sum_{i=1}^{N} R(i)$$

where N is the number of images.

$AR@0.5 : 0.95$ is calculated similarly to previously seen:

$$AR@[0.5 : 0.95](\%) = \left(\frac{1}{10}\sum_{i=0.5}^{0.95} AR(i)\right) \times 100$$

A high $AP@0.5$ result indicates that the model is good at correctly identifying a person when there is moderate overlap between the predicted and ground-truth bounding boxes. A high $AP@[0.5 : 0.95]$ indicates that the model performs well across a range of localisation strictness levels.

A high $AR@0.5 : 0.95$ indicates that the model is good at finding the relevant person within an image, across a range of overlap (IoU) thresholds.

Finally we used activation maps [71] to measure which features of input data were being recognised by the models we were training. Activation maps provide a heat map of the features activated by different layers of a neural network and help confirm that a model concentrates on the correct object it is trying to detect, rather than irrelevant features from the training data.

3.7.2 Measuring SAHI's impact on performance

To evaluate how adaptive slicing affected performance, we incorporated it into the inference pipeline of our best-performing model, rerunning the model on the test set [72]. We measured both AP and AR over the aforementioned intersection over unions (IoU) and the difference in frames per second (FPS). This comparison was used to determine if any potential decrease in FPS is an acceptable trade-off for the increase in precision that SAHI offered.

3.7.3 Evaluating the SAR simulation

Given that we could not obtain direct feedback from someone within a SAR department or run the tool on a real SAR mission, the simulation evaluation focused on determining the tool's functionality and usefulness. Because of this, we measured just one quantitative figure, the FPS of the output video produced by the system. Qualitative feedback from the simulation was discussed under three headers:

1) **System Functionality** – Is the system functioning as intended? This topic is explored in detail in Section 5.2.1, where we assess the system's operational effectiveness.
2) **Detection Capabilities & Usefulness** – Does the model function as intended? Can it detect individuals before the operator, thus aiding in identification? Are there noticeable deficiencies or limitations of the model under specific conditions or scenarios? These questions are addressed in Section 5.2.2
3) **Limitations** – Were there any other system constraints observed during the SAR simulation? These are discussed in Section 5.2.3

4 Results

4.0.1 Training

In this section we show the training and validation results produced by each respective model, showing that all models achieved a state of approximate convergence during the training phase. The training parameters of each model are shown in Table 7.

TABLE 7: Object Detection Architecture Training Parameters

Model	Learning Rate	Momentum	Optimiser	Weight Decay	Batch Size	Epochs
YOLOv8-nano	0.01	0.937	AdamW first 5 epochs, then SGD	0.0005	16	100
RTMDet-tiny	0.004	0.0002	AdamW	0.05	16	100
Faster R-CNN-R50-FPN	0.02	0.9	AdamW	0.0001	16	100

Fig. 11: RTM-Det training loss

Fig. 12: YOLOv8-nano training loss

Fig. 13: Faster R-CNN training loss

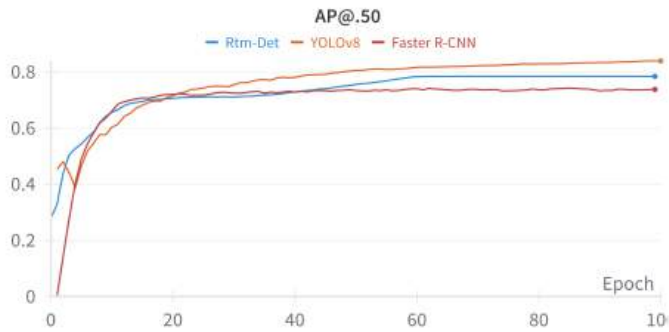

Fig. 14: Validation performance at every epoch: Average Precision (AP) at 0.5% IoU threshold

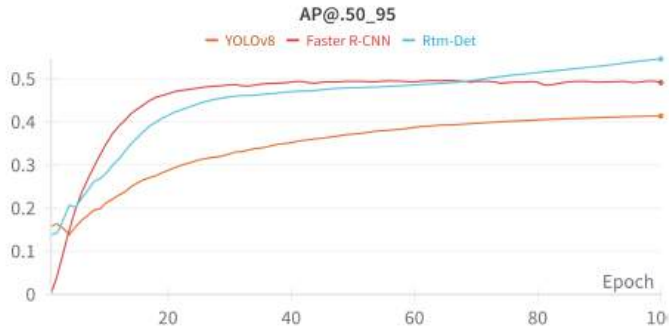

Fig. 15: Validation performance at every epoch: Average Precision (AP) with IoU threshold from 0.5 to 0.95

4.0.2 *Testing*

This section presents the performance metrics obtained from the test set for each corresponding model. From the results in Table 8 and Table 9 we concluded that the YOLOv8-nano model without slicing adaptive inference (SAHI) was the most appropriate for deployment in the SAR simulation. The evaluation of the results and the justification for choosing the YOLOv8-nano model are explained in Section 5.1.1.

TABLE 8: Model performance metrics on the test set, showing that YOLOv8-nano was the best-performing model on most evaluation metrics

Model	Size (pixels)	Parameters (M)	AP@0.5	AP@0.5:0.95	AR@0.5:0.95	FPS
YOLOv8-nano	640	**3.2**	**83**	42.2	**74.9**	**21.97**
RTMDet-tiny	640	4.88	78	**66**	48.8	10.32
Faster R-CNN-R50-FPN	800	41.34	74	50	53.7	1.43

TABLE 9: Performance comparison of YOLOv8-nano model with and without slicing adaptive inference (SAHI), showing that SAHI offered a performance increase but an unreasonable trade-off in inference speed

Model	AP@0.5	AP@0.5:0.95	AR@0.5:0.95	FPS	Slice Size
YOLOv8-nano	83	42.2	74.9	**21.97**	**N/A**
YOLOv8-nano-SAHI	**84.3**	**57**	74.9	3.2	256

Fig. 16: Sample of YOLOv8 Predictions on test set, Prediction (Red) vs Ground Truth (Green)

4.0.3 Simulation

This section shows screenshots taken from the setup of the system and performance in simulation runs. Figure 17 shows an example detection in a simulation run, and Appendix A.13 provides further images from SAR simulation experiments.

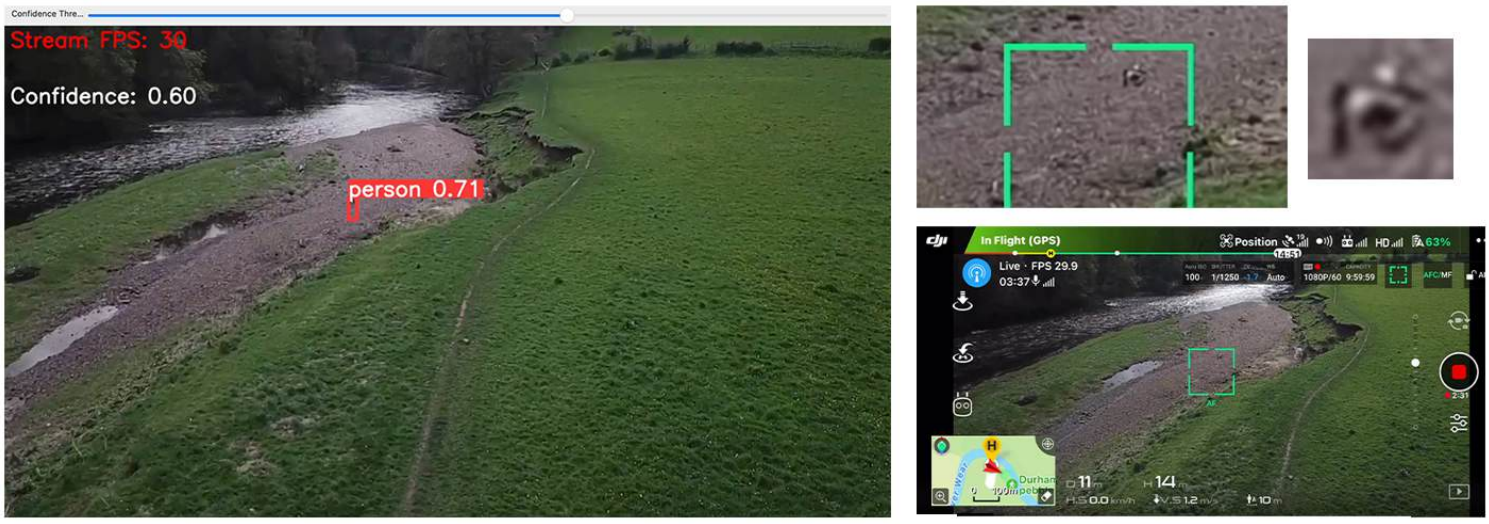

Fig. 17: Search and rescue simulation screenshots: Left shows the computer's detection of a person in the video feed. The right presents the drone controller's view, with a zoomed inset showing the person's location.

5 EVALUATION

5.1 Model Training and Testing

In this section, we provide a comprehensive evaluation of the results of our experiments. Section 5.1.1 discusses the performance of the three trained object detection models and why the YOLOv8-nano model was chosen as the best performing model. Section 5.1.2 evaluates the trade-off in performance SAHI offered to the YOLOv8-nano model. Finally, Section 5.2 discusses the final system's performance in a simulated SAR operation.

5.1.1 General Model Performance

Commenting on the general performance of the models, Table 8 shows that the algorithms proposed in this paper did very well on the composite dataset. The algorithms—Faster R-CNN, YOLOv8-nano, and RTMDet—achieved high AP scores of 74%, 83%, and 78% respectively, at an IoU threshold of 50% (AP@0.5) during testing. However, when the AP was measured over multiple IoU thresholds, from 0.5 to 0.95, in steps of 0.05, the models performed significantly worse on the test set. Figure 15 shows the AP@0.5:0.95 for each algorithm during the training process, detailing that poor AP@0.5:0.95 results were not limited to the test set. This likely occurred because, in detecting small objects where bounding boxes are also smaller, minor misalignments with the ground truth can cause significant discrepancies in IoU measures, and the AP@0.5:0.95 metric uses more strict thresholds. The average recall, measured as AR@0.5:0.95, was exclusively high for the YOLOv8-nano model, which achieved a 74.9% AR@0.5:0.95 on the test set.

Table 8 also shows that the YOLOv8-nano model outperformed the other base models, in the majority of evaluation metrics. The base YOLOv8-nano model boasted the highest AP@0.5 on the test set whilst being the smallest model (just 3.2 million parameters) and running at close to 22 FPS, the only model to achieve near real-time performance. As such, the YOLOv8-nano model was chosen as the best performing base model, able to detect people in a wide range of remote scenarios and capable of running in close to real-time.

5.1.2 Integration of SAHI

The integration of SAHI into the YOLOv8-nano model showed promising performance improvements. We opted for a slice size of 256x256, considering that smaller slices could hinder inference speed. However, as detailed in Table 9, even with this slice size, the reduction in frames per second (FPS) during device inference was significant. This reduction led us to determine that incorporating SAHI into the inference pipeline was impractical for deployment on lower-compute devices in real-time settings. Despite this, our research highlights that SAHI significantly enhances the system's detection capabilities, especially for distant person detection. Thus, if a device with adequate computing power is available, incorporating SAHI is highly recommended. The integration of SAHI with the YOLOv8-nano model led to a remarkable 15% increase in AP@0.5:0.95 on the test set, demonstrating a substantial enhancement in performance.

5.2 Performance in Simulation

5.2.1 System Functionality

The system performed well during the simulation offering a straightforward setup; the operator simply ran a Python script that processed video frames from an RTMP (Real-Time Messaging Protocol) server and input the server's IP address into the DJI app. For wider accessibility, integrating these steps into a single executable would make operation simpler for users without technical expertise. Our system is compatible with any drone capable of streaming RTMP data, not just those using the DJI app.

During the simulation, the streaming rate fluctuated between 20-30 FPS, with a noticeable lag of 1-5 seconds between the controller's view and the laptop stream. Despite this, the system consistently detected the lost person in multiple trials. Figure 17 illustrates a notable detection where the model identified a person lying on a pebble beach. Additional details and images from the simulation setup and results are available in Appendix A.13.

We included a slider in our Python program to adjust the model's confidence threshold τ for detecting a person. The initial setting of $\tau = 0.6$ (60% confidence) yielded satisfactory results with minimal false detections, and we chose not to adjust this setting further. The system functioned as anticipated, processing streamed data effectively. Both the computer and the phone maintained a stable connection to the same Wi-Fi network, which is crucial as the system's performance hinges on a robust Wi-Fi connection. A weaker connection led to reduced stream quality and visible artifacts; however, the model's performance remained robust, likely benefiting from similar scenarios addressed during data augmentation in training.

5.2.2 Detection Capabilities & Usefulness

The model demonstrated remarkable performance during the simulation, frequently identifying the lost person before they were visible to the operator on the phone controller's screen. The tests were conducted in a variety of settings, including WiSAR terrains with backgrounds of grass, greenery, pebbles, and grey shades, where the model performed well in all scenarios.

However, the model occasionally failed to detect the full body of a person, relying instead on the legs for identification. This scenario was observed in multiple simulation runs, with Figure 18 showcasing instances where only the legs were recognised. Such findings underscore the unique challenges and insights that emerge from real-world application testing.

Additionally, Figure 19 presents activation maps from the YOLOv8-nano model used during the tests. These maps, produced prior to deployment, did not initially reveal the tendency to overlook upper body parts, highlighting an area for further investigation and model refinement.

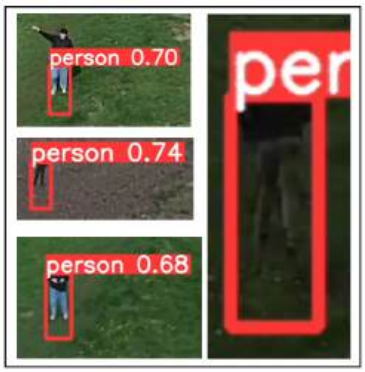

Fig. 18: Sample of detections during simulation, where only the legs of individuals were detected

Fig. 19: Example attention maps from test set inference. Original images on the left, corresponding attention heatmaps on the right

5.2.3 Limitations

There were several limitations and notable issues with the proposed system, most of which could be easily addressed. Firstly, the laptop screen and output detections were sometimes difficult to visualise in daylight conditions. Additionally, for detections of persons at a distance, the detection boxes could be upscaled or enhanced to improve visibility for the operator. Lastly, it is important to emphasize that this system has only been deployed and benchmarked on a single device. Further testing on various hardware configurations is necessary to fully assess the system's capabilities and limitations.

6 Conclusion

This paper introduces a lightweight, real-time person detection system designed for search and rescue (SAR) operations, which can be deployed on consumer laptops/devices and is compatible with various drone setups. The system leverages a computer vision task via object detection to detect missing persons from drone camera feeds in real time. Initially, an extensive evaluation of existing SAR datasets was conducted to identify terrain biases, leading to the creation of a novel synthetic dataset tailored for person detection in desert and mountainous terrains. This synthetic dataset was integrated with existing datasets to form a composite dataset for remote SAR applications. To build a computer vision model capable of real-time person detection on hardware with limited GPU resources, multiple state-of-the-art object detection algorithms were trained, and their performances were compared. The YOLOv8-nano model was shown to be the most effective, achieving an average precision (AP@0.5) of 83% on the test set and operating at near real-time speeds on the selected hardware. Further experiments were conducted to enhance the YOLOv8-nano model, including an evaluation of slicing adaptive inference (SAHI); however, it proved too resource-intensive for practical real-time deployment. The finalised YOLOv8-nano model was deployed on a consumer laptop as part of the proposed detection system, enabling a drone to stream video data via the Real-Time Messaging Protocol (RTMP) to a ground-based device. This setup allowed the deployed model to process live video streams from a drone-mounted camera and deliver detections to a ground SAR operator for visualisation. The effectiveness of the model and the system was demonstrated during a SAR simulation, where it successfully aided in identifying a missing person. The paper achieves its objectives, validating the system's success and potential utility in SAR operations.

The potential for future work on this project is extensive. Additional experiments are required to thoroughly assess the regions of interest within the feature extraction pipeline of the final model, to evaluate system performance in actual SAR operations.

There is a vast scope for future work which could extend this project. Further experiments are to be done evaluating the final model regions of interest within the feature extraction pipeline, the system's performance in real SAR operations, and to test the models on various hardware benchmarks.

We recognise that in specific SAR scenarios, such as those in alpine environments, a lost individual may be completely obscured by snow [73], which can severely limit the effectiveness of our current system. To address this, an important future direction would be to adapt and retrain our model for compatibility with infrared camera systems. This adaptation would ensure that our system remains functional and effective in SAR operations that utilise infrared

technology, thus broadening the scope and applicability of our approach to include a wider range of search and rescue contexts.

Since the beginning of this project, there have been significant advancements in open vocabulary object detection; a work that enables models to identify objects based on text prompts. One exciting avenue for future work involves incorporating the newly developed 'YOLO-World' model [74], a state-of-the-art, real-time open-vocabulary detection system, into our person detection framework. This enhancement could be extremely beneficial in SAR missions, enabling operators to search based on specific descriptions of missing individuals. However, the challenge addressed in this paper remains; integrating this advanced technology into cost-effective hardware solutions that can be widely distributed to SAR teams globally. This step is crucial for ensuring the accessibility and practical application of these enhancements in real-world search and rescue operations.

REFERENCES

[1] M. Lyu, Y. Zhao, C. Huang, and H. Huang, "Unmanned aerial vehicles for search and rescue: A survey," *Remote Sensing*, vol. 15, no. 13, p. 3266, 2023. [Online]. Available: https://doi.org/10.3390/rs15133266

[2] C. P. Catapult, "Using drones in search and rescue operations," Connected Places Catapult, March 2022. [Online]. Available: https://cp.catapult.org.uk/news/using-drones-in-search-and-rescue-operations/

[3] S. Rhode. (2018) Drone search-and-rescue study reveals potential and limits. [Online]. Available: https://www.aopa.org/news-and-media/all-news/2018/october/01/drone-study-reveals-potential-and-limits

[4] S. Gotovac, D. Zelenika, Ž. Marušić, and D. Božić-Štulić, "Visual-based person detection for search-and-rescue with uas: Humans vs. machine learning algorithm," *Remote Sensing*, vol. 12, no. 20, p. 3295, 2020.

[5] K. Akshatha, A. K. Karunakar, S. B. Shenoy, A. K. Pai, N. H. Nagaraj, and S. S. Rohatgi, "Human detection in aerial thermal images using faster r-cnn and ssd algorithms," *Electronics*, vol. 11, no. 7, p. 1151, 2022.

[6] M. Rizk, F. Slim, and J. Charara, "Toward ai-assisted uav for human detection in search and rescue missions," in *2021 International Conference on Decision Aid Sciences and Application (DASA)*. IEEE, 2021, pp. 781–786.

[7] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling, "Detection and tracking meet drones challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7380–7399, 2021.

[8] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: Object detection and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 370–386.

[9] K. Sandvik, "African drone stories," *Behemoth A Journal on Civilization*, vol. 8, no. 2, 2015.

[10] E. Tatsidou, C. Tsiamis, E. Karamagioli, G. Boudouris, A. Pikoulis, E. Kakalou, and E. Pikoulis, "Reflecting upon the humanitarian use of unmanned aerial vehicles (drones)," *Swiss Medical Weekly*, vol. 149, no. 1314, pp. w20 065–w20 065, 2019.

[11] I. Daou, "Tapping on the current and future potential of drones in saving human lives."

[12] I. Martinez-Alpiste, P. Casaseca-de-la Higuera, J. Alcaraz-Calero, C. Grecos, and Q. Wang, "Benchmarking machine-learning-based object detection on a uav and mobile platform," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[13] L. Johnson, "An introduction to mountain search and rescue," *Emergency Medicine Clinics*, vol. 22, no. 2, pp. 511–524, 2004.

[14] A. Peake, J. McCalmon, Y. Zhang, B. Raiford, and S. Alqahtani, "Wilderness search and rescue missions using deep reinforcement learning," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2020, pp. 102–107.

[15] J. Carneiro, "Maritime search and rescue," *IETE Technical Review*, vol. 5, no. 3, pp. 111–114, 1988.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[17] Y. Pang, Y. Yuan, X. Li, and J. Pan, "Efficient hog human detection," *Signal processing*, vol. 91, no. 4, pp. 773–781, 2011.

[18] P. Blondel, A. Potelle, C. Pégard, and R. Lozano, "Fast and viewpoint robust human detection for sar operations," in *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, 2014, pp. 1–6.

[19] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. von Stryk, S. Roth, and B. Schiele, "Vision based victim detection from unmanned aerial vehicles," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1740–1747.

[20] S. Sambolek and M. Ivasic-Kos, "Automatic person detection in search and rescue operations using deep cnn detectors," *IEEE Access*, vol. 9, pp. 37 905–37 922, 2021.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[22] C. Lyu, W. Zhang, H. Huang, Y. Zhou, Y. Wang, Y. Liu, S. Zhang, and K. Chen, "Rtmdet: An empirical study of designing real-time object detectors," *arXiv preprint arXiv:2212.07784*, 2022.

[23] S. Gupta and D. T. U. Devi, "Yolov2 based real time object detection," *Int. J. Comput. Sci. Trends Technol. IJCST*, vol. 8, pp. 26–30, 2020.

[24] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Search and rescue operation using uavs: A case study," *Expert Systems with Applications*, vol. 178, p. 114937, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742100378X

[25] S. Sambolek and M. Ivasic-Kos, "Automatic person detection in search and rescue operations using deep cnn detectors," *Ieee Access*, vol. 9, pp. 37 905–37 922, 2021.

[26] B. Mishra, D. Garg, P. Narang, and V. Mishra, "Drone-surveillance for search and rescue in natural disaster," *Computer Communications*, vol. 156, pp. 1–10, 2020.

[27] M. Rizk and I. Bayad, "Human detection in thermal images using yolov8 for search and rescue missions," in *2023 Seventh International Conference on Advances in Biomedical Engineering (ICABME)*. IEEE, 2023, pp. 210–215.

[28] L. Du, R. Zhang, and X. Wang, "Overview of two-stage object detection algorithms," in *Journal of Physics: Conference Series*, vol. 1544, no. 1. IOP Publishing, 2020, p. 012033.

[29] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[30] H. Zhang, C. Hao, W. Song, B. Jiang, and B. Li, "Adaptive slicing-aided hyper inference for small object detection in high-resolution remote sensing images," *Remote Sensing*, vol. 15, no. 5, p. 1249, 2023.

[31] S. H. Alsamhi, A. V. Shvetsov, S. Kumar, S. V. Shvetsova, M. A. Alhartomi, A. Hawbani, N. S. Rajput, S. Srivastava, A. Saif, and V. O. Nyangaresi, "Uav computing-assisted search and rescue mission framework for disaster and harsh environment mitigation," *Drones*, vol. 6, no. 7, p. 154, 2022.

[32] L. A. Varga, B. Kiefer, M. Messmer, and A. Zell, "Seadronessee: A maritime benchmark for detecting humans in open water," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022, pp. 2260–2270.

[33] S. Sambolek and M. Ivasic-Kos, "Search and rescue image dataset for person detection - sard," 2021. [Online]. Available: https://dx.doi.org/10.21227/ahxm-k331

[34] D. Broyles, C. R. Hayner, and K. Leung, "Wisard: A labeled visual and thermal image dataset for wilderness search and rescue," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9467–9474.

[35] D. Božić-Štulić, Ž. Marušić, and S. Gotovac, "Deep learning approach in aerial imagery for supporting land search and rescue missions," *International Journal of Computer Vision*, vol. 127, no. 9, pp. 1256–1278, 2019.

[36] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolo," Ultralytics, Jan. 2023, if you use this software, please cite it using the metadata from this file. [Online]. Available: https://github.com/ultralytics/ultralytics

[37] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[38] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, pp. 850–855, 2006.

[39] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[40] H. M. Khan and C. Yunze, "Ship detection in sar image using yolov2," in *2018 37th Chinese control conference (CCC)*. IEEE, 2018, pp. 9495–9499.

[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[42] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[43] B. Valarmathi, J. Kshitij, R. Dimple, N. Srinivasa Gupta, Y. Harold Robinson, G. Arulkumaran, T. Mulu *et al.*, "Human detection and action recognition for search and rescue in disasters using yolov3 algorithm," *Journal of Electrical and Computer Engineering*, vol. 2023, 2023.

[44] NVIDIA Corporation, "Nvidia jetson tx1," NVIDIA Developer, 2015, embedded AI Computing Device. [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx1

[45] ——, "Nvidia jetson xavier nx," NVIDIA Developer, 2020, embedded AI Computing Device. [Online]. Available: https://developer.nvidia.com/embedded/jetson-xavier-nx

[46] P. Adarsh, P. Rathi, and M. Kumar, "Yolo v3-tiny: Object detection and recognition using one stage improved model," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 687–694.

[47] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[48] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[49] A. Name], "From yolo to yolov8: Tracing the evolution of object detection algorithms," https://medium.com/nerd-for-tech/from-yolo-to-yolov8-tracing-the-evolution-of-object-detection-algorithms-eaed9a982ebd, 2023, accessed: [Access Date].

[50] S. Liu, H. Zhou, C. Li, and S. Wang, "Analysis of anchor-based and anchor-free object detection methods based on deep learning," in *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2020, pp. 1058–1065.

[51] I. P. Sary, S. Andromeda, and E. U. Armin, "Performance comparison of yolov5 and yolov8 architectures in human detection using aerial images," *Ultima Computing: Jurnal Sistem Komputer*, vol. 15, no. 1, pp. 8–13, 2023.

[52] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[53] KeyLabs, "Yolov8 vs. ssd: Choosing the right object detection model," https://keylabs.ai/blog/yolov8-vs-ssd-choosing-the-right-object-detection-model/, 2024, accessed: 2024-04-20.

[54] S. Liu, H. Zou, Y. Huang, X. Cao, S. He, M. Li, and Y. Zhang, "Erf-rtmdet: An improved small object detection method in remote sensing images," *Remote Sensing*, vol. 15, no. 23, p. 5575, 2023.

[55] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[56] M. Lyu, Y. Zhao, C. Huang, and H. Huang, "Unmanned aerial vehicles for search and rescue: A survey," *Remote Sensing*, vol. 15, no. 13, p. 3266, 2023.

[57] Q. Wang, J. Gao, W. Lin, and Y. Yuan, "Learning from synthetic data for crowd counting in the wild," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8198–8207.

[58] J. Yu, D. Farin, C. Krüger, and B. Schiele, "Improving person detection using synthetic training data," in *2010 IEEE International Conference on Image Processing*. IEEE, 2010, pp. 3477–3480.

[59] "Grand theft auto v," Video game, 2013, available on PlayStation 3, PlayStation 4, Xbox 360, Xbox One, and PC.

[60] Y.-T. Hu, H.-S. Chen, K. Hui, J.-B. Huang, and A. G. Schwing, "Sail-vos: Semantic amodal instance level video object segmentation-a synthetic dataset and baselines," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3105–3115.

[61] S. Lin, K. Wang, X. Zeng, and R. Zhao, "Explore the power of synthetic data on few-shot object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 638–647.

[62] E.-J. Lee, D. M. Conover, S. S. Bhattacharyya, H. Kwon, J. Hill, and K. Evensen, "Validation of object detection in uav-based images using synthetic data," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, vol. 11746. SPIE, 2021, pp. 584–601.

[63] N. Clement, A. Schoen, A. Boedihardjo, and A. Jenkins, "Synthetic data and hierarchical object detection in overhead imagery," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 20, no. 4, pp. 1–20, 2024.

[64] R. Team, "RoboFlow: A Platform for Computer Vision Development," https://www.robolflow.com, 2023, available online: https://www.robolflow.com (accessed on April 6, 2024).

[65] PaperSpace Co., "Paperspace: Cloud gpu computing platform," https://www.paperspace.com, 2024, accessed: 2024-04-06.

[66] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[67] ONNX Contributors, "Open neural network exchange," https://github.com/onnx/onnx, 2023, accessed: 2024-04-06.

[68] D. Chaurasia and B. Patro, "Real-time detection of birds for farm surveillance using yolov7 and sahi," in *2023 3rd International Conference on Computing and Information Technology (ICCIT)*. IEEE, 2023, pp. 442–450.

[69] Z. Fan, B. Guo, and J. Hou, "Implementation of a drone-based video streamer," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. Springer, 2017, pp. 67–74.

[70] Amazon Web Services, "Live streaming from specialized live cameras and drones using rtmp to amazon ivs," https://aws.amazon.com/blogs/media/live-streaming-from-specialized-live-cameras-and-drones-using-rtmp-to-amazon-ivs/, April 2023, accessed: 2024-04-06.

[71] R. Satish, "Eigencam for yolo v8 interpretability," https://github.com/rigvedrs/YOLO-V8-CAM, 2023.

[72] F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing aided hyper inference and fine-tuning for small object detection," *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 966–970, 2022.

[73] F. Fruehauf, A. Heilig, M. Schneebeli, W. Fellin, and O. Scherzer, "Experiments and algorithms to detect snow avalanche victims using airborne ground-penetrating radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2240–2251, 2009.

[74] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, "Yolo-world: Real-time open-vocabulary object detection," *arXiv preprint arXiv:2401.17270*, 2024.

7 APPENDIX

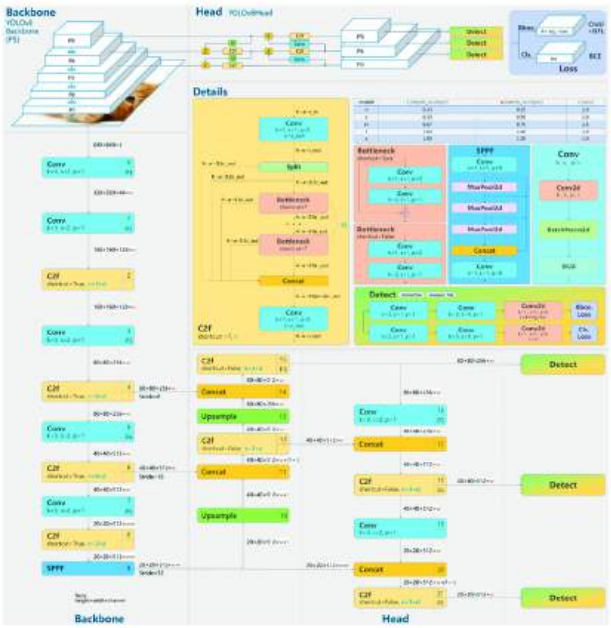

Fig. A.1: Diagram showing the YOLOv8 architecture [36]

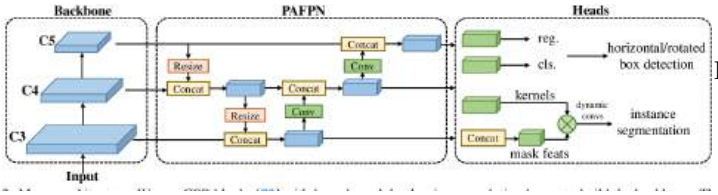

Fig. A.2: Diagram showing the RTMDet architecture [22]

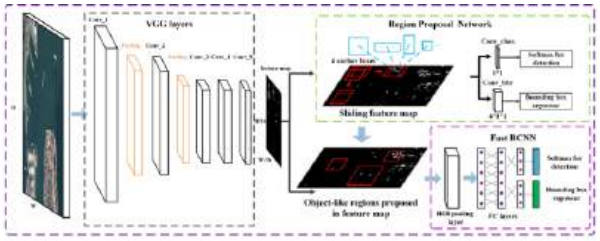

Fig. A.3: Diagram showing the Faster-RCNN architecture [21]

Fig. A.4: Images showing annotated SARD Dataset [32]

Fig. A.5: Images showing annotated SeeDroneSea Dataset [33]

Fig. A.6: Images showing annotated WiSARD Dataset [34]

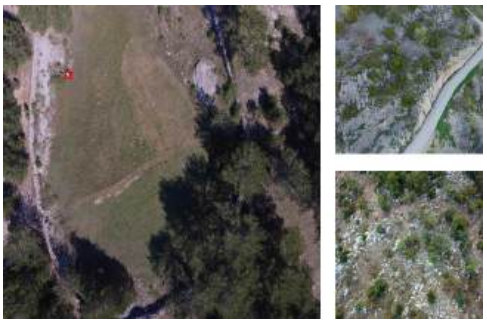

Fig. A.7: Images showing annotated HERIDAL Dataset [35]

Fig. A.8: Images showing annotated synthetic dataset 'GTASar'

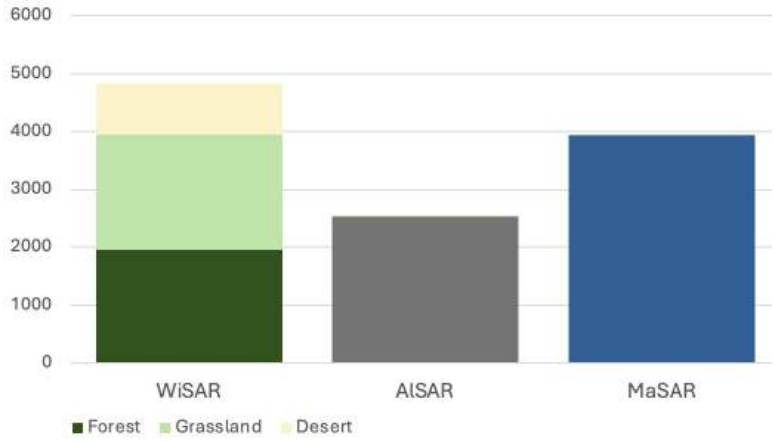

Fig. A.9: Distribution of SAR Labels in the Training Split of the final dataset

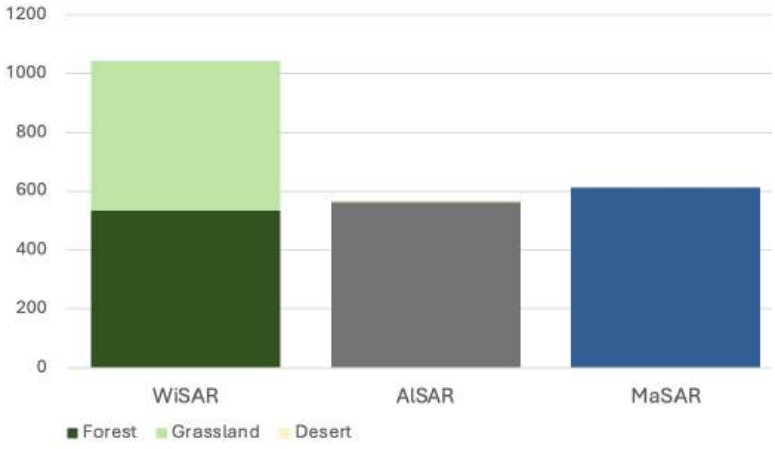

Fig. A.10: Distribution of SAR Labels in the Validation Split of the final dataset

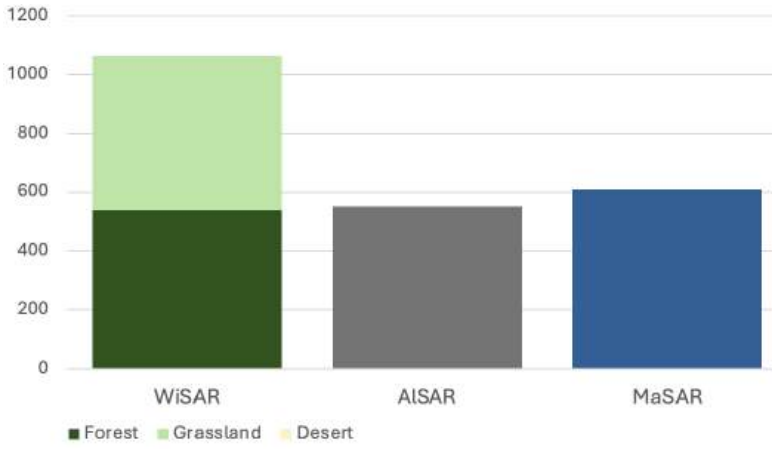

Fig. A.11: Distribution of SAR Labels in the Testing Split of the final dataset

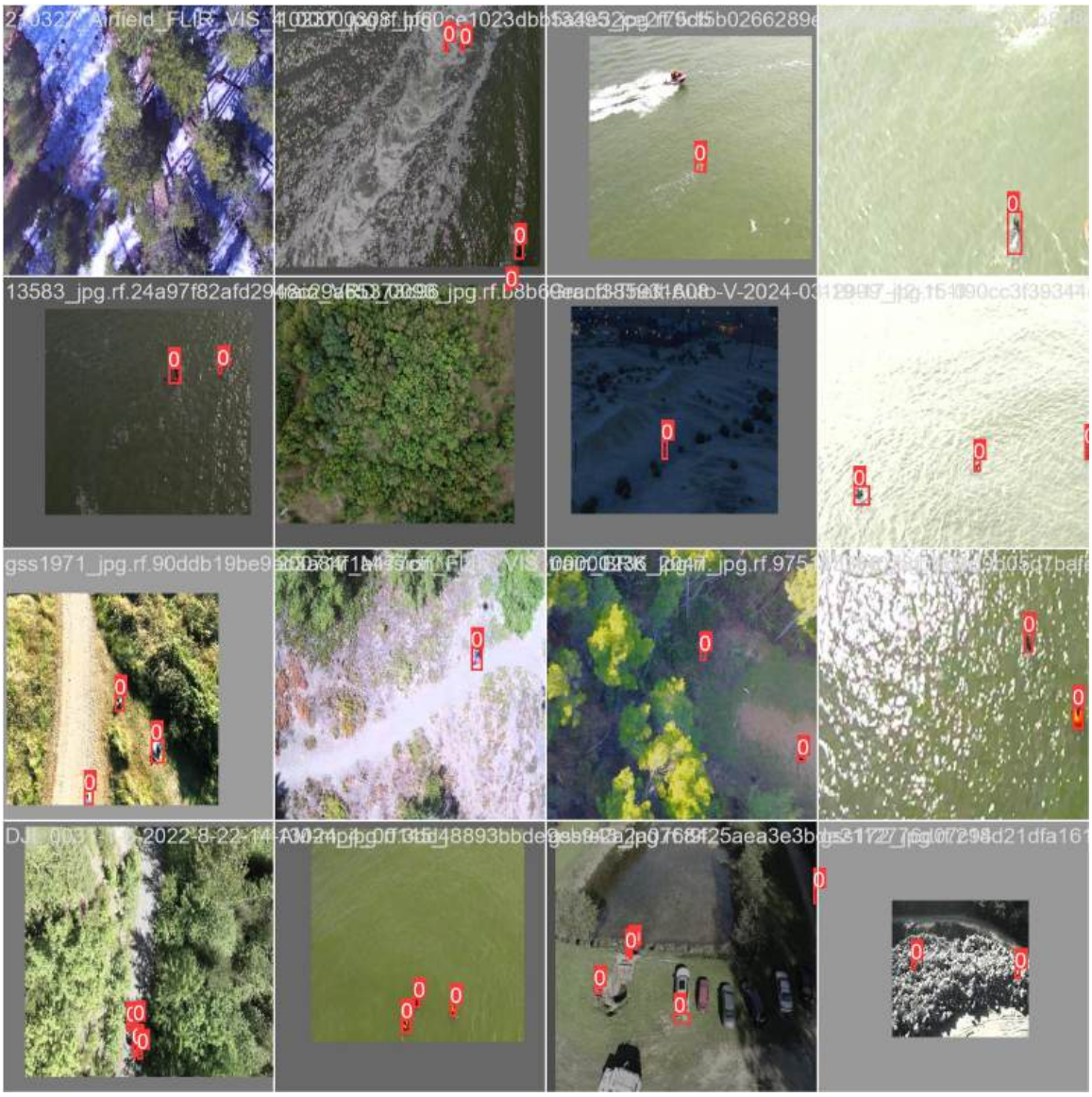

Fig. A.12: The final 16 images from the last set of the YOLOv8 training process (epoch 99) showing detections of class 0, which represents the 'person' category.

Fig. A.13: A collection of images taken from the SAR simulation run, and system testing