UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Construction of the object detection model for identifying mines and artillery shells

Author:
Nazar DOBROVOLSKYY

Supervisor:
Ph.D. Taras FIRMAN

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences and Information Technologies
Faculty of Applied Sciences

Lviv 2023

Declaration of Authorship

I, Nazar DOBROVOLSKYY, declare that this thesis titled, "Construction of the object detection model for identifying mines and artillery shells" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Still, if you will not fight for the right when you can easily win without bloodshed; if you will not fight when your victory will be sure and not too costly; you may come to the moment when you will have to fight with all the odds against you and only a precarious chance of survival. There may even be a worse case. You may have to fight when there is no hope of victory, because it is better to perish than to live as slaves."

Winston Churchill

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Construction of the object detection model for identifying mines and artillery shells

by Nazar DOBROVOLSKYY

Abstract

Among the devastation of the Russian-Ukrainian war, the conquered territories became a breeding ground for unexploded mines and ammunition. The indiscriminate use of these deadly weapons has caused countless civilian casualties and hampered recovery efforts in the affected regions. The urgent need to clear these areas requires effective and reliable detection of hidden mines and ordnance, which is challenging for human sappers due to the complexity and variability of the terrain and associated hazards.

In this thesis, we propose solving this problem by building an object detection model using the YOLOv7 algorithm. We pre-collected and processed a large image dataset, created annotations for mines and projectiles, and trained and evaluated a model on this dataset using standard evaluation metrics. Our results show that our model provides high accuracy and efficiency in detecting and locating mines and projectiles in various scenarios.

This research contributes to object detection and has practical implications for humanitarian demining operations. The model we developed integrates into ground remote robots for autonomous scanning and mapping of minefields, which reduces the risk of human casualties and accelerates the process of clearing territories.

This thesis highlights the need to utilize new technology to address the critical issues of post-conflict rebuilding and human security. We believe our effort will encourage future study and development in this subject, ultimately contributing to peace and stability in the afflicted countries.…

Acknowledgements

I want to express my sincere gratitude to the Armed Forces of Ukraine, which courageously defend the country's sovereignty and territorial integrity from the Russian Federation's aggression. Their sacrifices and dedication to the noble cause of freedom and democracy inspire me daily, and I am honored to contribute to their mission through this research.

I also sincerely appreciate Ukraine's State Emergency Service and the sappers who risk their lives to clear the area of mines and explosive weapons. Their hard efforts and expertise have saved numerous lives and enabled displaced people to return home safely.

I am deeply grateful to my supervisor, Taras Firman, for his guidance, support, and feedback throughout this research.

Finally, I want to thank Ukrainian Catholic University for giving me the academic tools and platform I needed to pursue my interest in science and technology. I feel delighted to contribute to Ukrainian Catholic University's reputation for excellence as a proud community member...

Contents

List of Figures

List of Tables

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
WSF	**W**hat (it) **S**tands **F**or
YOLO	**Y**ou **O**nly **L**ook **O**nce
mAP	**m**ean **A**verage **P**recision
AP	**A**verage **P**recision
IoU	**I**ntersection **o**ver **U**nion
TP	**T**rue **P**ositive
FP	**F**alse **P**ositive
FN	**F**alse **N**egative
TN	**T**rue **N**egative
CNN	**C**onvolutional **N**eural **N**etwork
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etwork
SSD	**S**ingle **S**hot **D**etector
COCO	**C**ommon **O**bjects (in) **CO**ntext
KITTI	**K**arlsruhe **I**nstitute (of) **T**echnology (and) **T**oyota **T**echnological **I**nstitute
CVAT	**C**omputer **V**ision **A**nnotation **T**ool
CSPDarkNet-53	**C**ross-**S**tage **P**artial **DarkNet-53**
VOC	**V**isual **O**bject **C**lasses
GPU	**G**raphics **P**rocessing **U**nit

Dedicated to the brave men and women who demine the territories of Ukraine

Chapter 1

Introduction

1.1 Background and Motivation

Russian aggression has left unexploded mines, artillery segments, and mined regions in Ukraine, posing a severe hazard to the people and recovery. According to the Ukrainian Ministry of Foreign Affairs [28], it is the most mined country in the world due to Russian military action. Mines and unexploded weapons pollute around 170-180 km of its territory. It might take up to 70 years to completely clear the Ukrainian land. Skilled sappers must find and destroy these deadly weapons without endangering people or losing their lives. However, searching for explosive objects and demining is complex and time-consuming, requiring new technologies to help sappers and speed up the process.

This bachelor's thesis aims to develop an object detection model based on the YOLOv7 algorithm, which can accurately and efficiently detect mines and artillery shells in various scenarios. The author collected data for training the model, contributing to the research's originality and uniqueness.

1.2 Research goals

The study aims to build an object detection model for identifying mines and artillery shells using the YOLOv7 algorithm. Specific goals are:

- Collect and process a large dataset of images of various mines and projectiles;
- Train and evaluate the YOLOv7 model on an annotated dataset using standard evaluation metrics;
- Analysis of the operation of the model on two microcomputers, Jetson Nano and Jetson Xavier, with subsequent integration of the model into ground remote robots for autonomous scanning and mapping of minefields.

1.3 Research

The model developed in this thesis can be integrated into ground remote robots for autonomous scanning and mapping of minefields, which reduces the risk of human casualties and accelerates the process of demining territories. Emphasis is placed on using advanced technologies to address the challenges of post-conflict reconstruction and human security. Object detection models can help minesweepers work and improve the safety of civilians living in affected areas.

Chapter 2

Literature Review

2.1 Introduction

This chapter brings to the forefront a literature review on YOLOv7, which provides timely, state-of-the-art detection of objects. YOLOv7 is an algorithm based on deep learning, which has become popular because of its very accurate and fast inferences. The YOLOv7 algorithm, its architecture, and the detection performance of objects are to be understood in this chapter.

In order to improve its performance, including data padding techniques and new loss functions, we are also reviewing recent work based on YOLOv7. We will also look at some trending datasets and programs used to detect objects and discuss the challenges faced by YOLOv7 and its successors.

2.2 Overview of Object Detection and CNN-based Methods

Object detection is an essential field of study in computer vision that has made significant progress in recent years. It includes identifying and creating bounding boxes around items of interest in an image or video. Because of their incredible accuracy and resilience, Convolutional Neural Networks (CNNs) have emerged as the dominant approach for object detection [10].

CNN-based methods for object detection can be broadly classified into two categories: two-stage methods and one-stage methods. Two-stage methods, such as Faster R-CNN [23], R-FCN [7], and Mask R-CNN [14], use a region proposal network to generate a set of candidate object locations, followed by a classification network to classify each object. One-stage methods, such as YOLO [22], SSD [19], and RetinaNet [16], perform object detection in a single shot without requiring a separate region proposal step.

YOLO (You Only Look Once) is a popular approach among one-stage methods due to its speed and accuracy [22]. YOLO divides an image into a grid of cells and predicts the probability of an object being present in each cell, along with the coordinates of the bounding box around the object. YOLO also predicts the class of the object using a multi-label classification approach. The penultimate version of YOLO, YOLOv7, has achieved state-of-the-art performance on several object detection benchmarks [4].

In addition to YOLO, other one-stage methods have been developed recently, such as SSD (Single Shot Detector) and RetinaNet. SSD uses a set of default bounding boxes of different aspect ratios and scales to detect objects [19]. RetinaNet introduces a new focal loss function that helps address the class imbalance problem in object detection [16].

CNN-based object detection methods have achieved remarkable progress in recent years, and there is still room for improvement. The following section will discuss the YOLOv7 algorithm and its contributions to object detection.

2.3 YOLOv7 overview

The YOLOv7 algorithm uses a "bag-of-freebies" approach, which incorporates several techniques shown to improve the performance of object detection models [4]. Specifically, YOLOv7 uses the compound scaling method, an expected parameterized model, and a compensatory loss for auxiliary heads. These techniques optimize various aspects of the architecture and training process to achieve better model accuracy and efficiency.

The compound scaling method involves scaling the model's depth, width, and resolution in a coordinated manner to optimize performance [5]. This approach enables the modeling to consider more sophisticated features while still being efficient. In order to minimize the number of hyperparameters and facilitate a more effective training system, the proposed parameterized model simplifies the architecture [25]. An assistant loss for an auxiliary head increases the model's accuracy by creating a new loss term to encourage it to discover more discriminative features [27].

Regarding architecture, YOLOv7 consists of a backbone network and a detection head. The backbone network is based on the CSPDarkNet-53 [3] architecture, which uses several convolutional layers with residual connections to extract the features of the input image [4]. In order to predict the location and classes of objects in an image, the detection head applies a series of convolution layers. Anchor boxes are used to improve object localization accuracy.

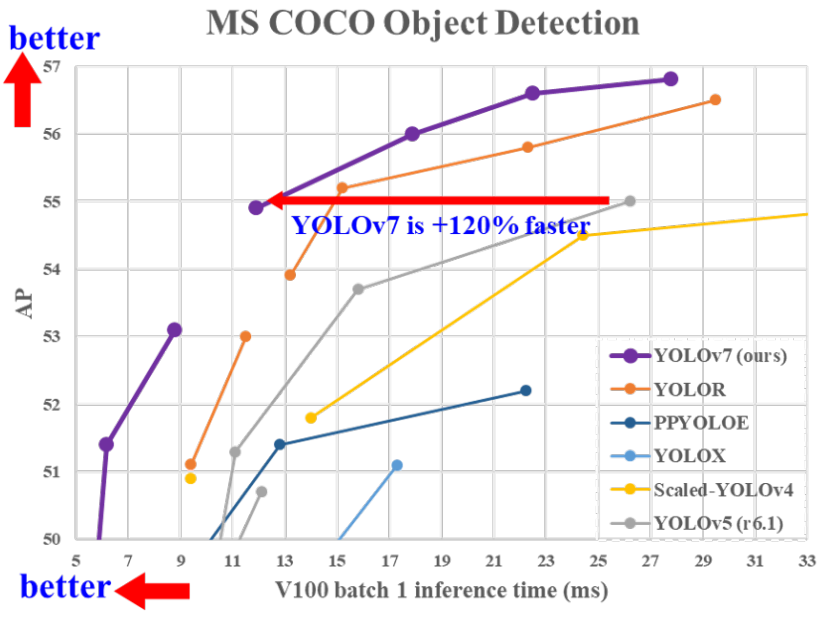

FIGURE 2.1: Comparison of object detection methods for the COCO dataset
Source: [4]

YOLOv7 achieves state-of-the-art performance on several benchmark datasets, including PASCAL VOC [12], COCO [17], and KITTI [13]. It achieves a mean average

precision (mAP) of 52.8% on COCO, significantly improving over previous state-of-the-art methods.

Recent work based on YOLOv7 includes the use of the algorithm for various applications, such as underwater target location [18] and damaged roads [20] and more. These studies demonstrate the versatility and effectiveness of YOLOv7 in real-world scenarios.

2.4 Advancements and Obstacles in Object Detection using YOLO

The problems linked to object detection, such as scale deviation, angle ratio, occlusion, and more, are addressed in Diwan's paper [8]. The paper points out that YOLO has tackled these problems by using a single neural network for classifying and finding objects in an image. In order to retrieve the features of an image, YOLO uses a series of convolutional layers that predict bounded boxes and class probability according to those characteristics.

The YOLO architecture comprises multiple convolution layers and a final completely interconnected layer. A convolutional layer extracts elements from an input image, and a fully interconnected layer performs the final prediction of object classes and bounding box coordinates. This paper provides a detailed overview of the Yolo architecture, composed of several layers used and their purpose.

Several improvements have been introduced by YOLOv2 in comparison to the original YOLO, such as anchor boxes, batch normalization, and multiscale training. Integrating a feature pyramid network and Darknet-53 [21] architecture into the YOLOv3 system has further enhanced these ideas. Finally, YOLOv4 introduced various optimization techniques, such as Mish activation and SPP-FCOS, which at the time of YOLOv4's introduction, exemplified state-of-the-art performance in object detection [3].

Several datasets used for training and evaluating the YOLO and its successors, such as COCO, Pascal VOC, or KITTI, are also covered in this paper. The COCO dataset is one of the most challenging datasets for detecting objects, and Pascal VOCs are widely used to benchmark object detection algorithms. KITTI is a set of datasets designed explicitly for Autonomous Driving applications, making it possible to evaluate object detection in realistic scenarios.

The paper also includes several examples of actual YOLO applications on the ground, such as drone detection, traffic monitoring, and more. The paper covers how YOLO can be used to identify and map objects simultaneously, making it an ideal instrument for surveillance or autonomous driving applications. Finally, Diwan's paper summarises various YOLO algorithms and related architectural successors, data sets, and applications. This paper emphasizes the object detection problem and illustrates how YOLO has dealt with it, resulting in an efficient and widely used method for detecting objects.

As the research in object detection algorithms and architectures continues, it is clear that YOLO-based models have proven promising for achieving real-time and accurate results. YOLOv7, with its bag-of-freebies trainable design, has pushed the state-of-the-art in real-time object detection [4]. Meanwhile, Tausif Diwan's study on YOLO's challenges, successors, datasets, and applications demonstrated that YOLO-based models had achieved exceptional performance in object detection tasks, particularly with the increasing availability of annotated datasets the advancements in hardware technology [8].

Finally, the literature review shows that YOLO-based models, particularly YOLOv7, have significantly advanced real-time object detection. Their architecture and design have made major improvements to today's state-of-the-art. Nonetheless, challenges remain, such as detecting small objects and dealing with occlusion. Researchers continually explore and develop new approaches and architectures to overcome these limitations.

Chapter 3

Methodology

3.1 Data description

The data collection process is a crucial step in developing an object detection model. This section will describe the data collection process for our model that identifies mines and artillery shells. The author of this thesis collected all data used in this research. This includes trips to landfills to collect photos of mines, visiting the Emergency Department in Chernivtsi, and getting access to data through UKROBORONSERVICE, which provided some of the data they had.

In total, 11 classes of mines and artillery shells were collected and summarized for this project. The classes include F-1, MM-120, MON-50, OG-9b, OZM-72, PFM-1, PMN (versions 1, 2 and 4), POM3-2m, RGO/RGN, TM-62 (with various activators: MVCh-62, MVN-72, MVN-80, MVP-62, and a case without an activator), and VOG-17m.

The data collection process involved taking photos of each class of us from various angles and distances to capture the necessary details for the model. The photos were taken using a 960 x 720 pixels digital camera. All photos from UKROBORONSERVICE were on the 4288 × 2848 resolution, with most being reduced to 1280 × 850. The data were collected indoors and outdoors to capture different lighting conditions.

Data collection also involved accessing data from UKROBORONSERVICE, including photos of mines and artillery shells. The data provided by UKROBORONSERVICE was used to supplement the data collected by the author of this thesis.

4,007 photos were collected for this project, with an average of 364 photos per class. The breakdown of the number of photos collected for each class is as follows:

Class	**Total Photos**
F-1	204
MM-120	435
MON-50	397
OG-9b	292
OZM-72	505
PRM-1	455
PMN	406
POM3-2m	519
RGO/RGN	446
TM-62	1,007
VOG-17m	123

TABLE 3.1: Class Distribution

Subclass	**Total Photos**
TM-62 without activator	150
TM-62 with MVZ-62 activator	328
TM-62 with MVCH-62 activator	149
TM-62 with MVN-72 activator	141
TM-62 with MVN-80 activator	149
TM-62 with MVP-62 activator	90

TABLE 3.2: Subclass Distribution of TM-62

Subclass	**Total Photos**
PMN 1	137
PMN 2	135
PMN 4	134

TABLE 3.3: Subclass Distribution of PMN

Subclass	**Total Photos**
RGO	270
RGN	176

TABLE 3.4: Subclass Distribution of RGO/RGN

The photos were then manually labeled using the CVAT [24] to create the dataset for training and validation. Sample data can be found on our GitHub repository [9], as well as the accompanying code that was used to build the model. To obtain the complete dataset, contact the authors of this thesis.

Also, to improve the model, we included negative data that do not contain mines and artillery shells. A total of 2,923 photos were collected for the data that do not include any mines and artillery shells.

Using null or negative data in training object detection models has become increasingly important in recent years. Negative data refers to images that do not contain the object of interest and are often used as a control group to ensure that the model is not biased toward specific features or backgrounds. In this study, we utilized a collection of image datasets containing negative examples, which we used as the -1 class in the YOLOv7 object detection model training.

Our primary goal was to detect and classify mines and artillery shells, which are notoriously difficult to identify because of their tiny size and distinctive look. To do so, we needed a broad set of photos to train the model to generalize to a wide range of scenarios. We selected to incorporate datasets collected by our hands from the environment - parks, stones, houses, roads, streets, and others. We also included photos of metal trash and different sorts of garbage that resembled the appearance of mines and artillery shells from the public access.

The datasets we used include those from Roboflow [1, 6], Kaggle [29, 2, 15], and GitHub [26]. The Roboflow datasets include pictures of metal scraps and objects, while the Kaggle datasets include pictures of waste, garbage, and domestic trash. The GitHub dataset we used contains pictures of various types of trash.

By including these datasets in the training data, we improved the model's ability to distinguish between objects of interest and the background. This was particularly important given the challenges of identifying mines and artillery shells in real-world scenarios, where various backgrounds and lighting conditions may exist.

3.2 The problem of data collection

Data collection for this project presented some challenges due to the nature of the objects being detected and the bureaucratization of obtaining data from military facilities. One of the main challenges was finding unexploded mines that could be used for training the object detection model. Due to safety concerns and legal restrictions, access to such mines is limited and requires special permits and approvals. As a result, samples were collected mainly by visiting landfills where decommissioned mines are stored.

Another challenge was the limited availability of data. The collected specimens are usually available only in one form, making their acquisition and processing complex. In addition, access to certain types of mines and artillery shells was restricted for security reasons. Even when access was granted, the bureaucratic process of obtaining the data was often time-consuming and involved numerous approvals and procedures.

Furthermore, the resolution of the collected photos varied significantly, which required preprocessing and augmentation to ensure that the images were suitable for training the object detection model. These challenges highlight the importance of properly planning and managing data collection for object detection models, especially when working with sensitive military materials.

3.3 Preprocessing and Augmentation of the Dataset

The dataset collected for this project required preprocessing and augmentation in order to improve the performance of the object detection model. The dataset was collected from various sources, including photos taken by the author at landfills and photos provided by UKROBORONSERVICE.

We used The CVAT tool to label the collected images. The labeled images were then used to train the YOLOv7 model. However, since the dataset was relatively small, data augmentation was necessary to increase the diversity of the images and improve the model's ability to generalize.

We used the Roboflow [11] platform to perform data augmentation. A total of 18,312 images were generated, split into the train, validation, and test sets of 16,020, 1,524, and 768 images, respectively. We applied the following data augmentation techniques to the images:

- Shear: $\pm 6^\circ$ Horizontal, $\pm 8^\circ$ Vertical;
- Saturation: Between -50% and $+50\%$;
- Brightness: Between -35% and $+35\%$;
- Exposure: Between -14% and $+14\%$;
- Blur: Up to 3.5px.

By augmenting the dataset with these techniques, the model could learn from a more diverse set of images, improving its performance on real-world data.

FIGURE 3.1: Sample data collected for the dataset

Chapter 4

Evaluation

The development of computer vision systems must include evaluating object detection models. In this chapter, we outline the assessment measures we utilized to rate the effectiveness of our YOLO-based object identification model. Measurements of the model's object detection capabilities and prediction accuracy compared to ground truth annotations are the two main objectives of the evaluation.

4.1 Mean Average Precision (mAP)

The mean average precision is a metric that is commonly used to evaluate object detection algorithms. It combines precision and recall values over various thresholds and is often reported at different intersection-over-union (IoU) values. In general, a higher mAP indicates better performance. The formula for calculating mAP is:

$$\text{mAP} = \frac{1}{n} \sum_{i=1}^{n} AP_i \tag{4.1}$$

where n is the number of classes, and AP_i is the average precision for class i. Average precision (AP) is calculated by computing the area under the precision-recall curve for a given class.

4.2 Average Precision (AP)

Average precision is a metric used to evaluate the performance of object detection algorithms for a specific class. It is calculated by computing the area under the precision-recall curve for a given class. The formula for calculating AP is:

$$\text{AP} = \int_0^1 p(r) \mathrm{d}r \tag{4.2}$$

where $p(r)$ is the precision at a given recall value r.

4.3 Intersection over Union (IoU)

IoU measures the overlap between the predicted and ground truth bounding boxes. To calculate IoU in YOLO, determine if the predicted bounding box and ground truth bounding box intersect. The formula for calculating IoU is:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{4.3}$$

4.4 Possible model outputs

- **True Positive (TP):** The model correctly detects an object that exists in the ground truth.
- **False Positive (FP):** The model detects an object that does not exist in the ground truth.
- **False Negative (FN):** The model fails to detect an object that exists in the ground truth.
- **True Negative (TN):** The model correctly identifies that there is no object in the region of interest where there is actually no object in the ground truth.

4.5 Recall

The recall is a metric that measures the percentage of ground truth objects that were correctly detected by the object detection algorithm. The formula for calculating recall is:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.4}$$

We used the mean average precision (mAP) metric to evaluate our YOLO model with an IoU threshold of 0.5 and 0.95, precision, and recall. These metrics provide a comprehensive assessment of the detection performance of the model. Based on our analysis of the results, we have focused on reporting the values of mAP at 0.5 and 0.95, as well as precision and recall, as they clearly indicate the model's ability to detect objects accurately.

4.6 Precision

Precision is a metric that measures the percentage of detections made by the object detection algorithm that was correct. The formula for calculating precision is:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.5}$$

Chapter 5

Experiments

In this chapter, we describe a series of experiments conducted to optimize the performance of the YOLOv7 object detection model for our use case. We aim to identify the best hyperparameters for the YOLOv7 model by training it with different configurations and evaluating its performance on a validation dataset.

The YOLOv7 architecture was used, specifically the `yolov7.yaml` configuration file for our experiments because it has been shown to provide state-of-the-art performance on object detection tasks. YOLOv7 offers a good balance between speed and accuracy, making it suitable for our use case.

We evaluated the YOLOv7 model on the Nvidia A100 40GB GPU using 20 workers and a batch size of 20. We used an image size of 960x720 pixels for training and validation. We selected the A100 GPU for its high performance, which allowed us to train the model efficiently.

The model was trained for 20 epochs with different hyperparameters for each experiment. The hyperparameters we varied included learning rate, momentum, weight decay, and others. We ran ten experiments with different hyperparameters to identify the best configuration for our use case.

The outcomes of our research and the performance of the YOLOv7 model with various hyperparameters are discussed in the following sections. Plots and visuals are also included to aid in understanding the model's performance. We aim to identify the best configuration of the YOLOv7 model for our use case, which we will use to train a final model.

The following tables present each experiment's hyperparameters that are used. They are numbered sequentially, starting from Experiment 1.

Hyperparameters	**Exp 1**	**Exp 2**	**Exp 3**	**Exp 4**	**Exp 5**	**Exp 6**	**Exp 7**	**Exp 8**	**Exp 9**	**Exp 10**
lr0	0.001	0.01	0.005	0.01	0.001	0.001	0.01	0.01	0.01	0.01
lrf	0.2	0.1	0.05	0.1	0.2	0.1	0.1	0.1	0.2	0.01
momentum	0.9	0.95	0.9	0.95	0.9	0.9	0.937	0.937	0.937	0.937
weight_decay	0.0001	0.0005	0.0001	0.0005	0.001	0.0001	0.0005	0.0005	0.0005	0.0005
warmup_epochs	3.0	4.0	5.0	4.0	2.0	2.0	3.0	3.0	3.0	3.0
warmup_momentum	0.6	0.7	0.9	0.7	0.6	0.8	0.8	0.8	0.8	0.8
warmup_bias_lr	0.05	0.1	0.2	0.1	0.2	0.2	0.1	0.1	0.1	0.1
box	0.1	0.05	0.1	0.05	0.1	0.1	0.05	0.05	0.05	0.05
cls	0.5	0.8	0.5	0.8	0.6	0.8	0.3	0.3	0.3	0.5
cls_pw	0.8	1.0	2.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0
obj	1.0	0.8	0.5	0.8	0.5	0.8	0.7	0.7	0.7	1.0
obj_pw	1.0	1.0	2.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0
iou_t	0.2	0.3	0.25	0.3	0.2	0.4	0.2	0.2	0.2	0.2
anchor_t	4.0	3.0	5.0	3.0	5.0	3.5	4.0	4.0	4.0	4.0
fl_gamma	0.0	1.0	1.0	1.0	2.0	2.0	0.0	0.0	0.0	0.0
hsv_h	0.02	0.02	0.03	0.02	0.03	0.03	0.015	0.015	0.015	0.015
hsv_s	0.6	0.6	0.5	0.6	0.5	0.8	0.7	0.7	0.7	0.7
hsv_v	0.4	0.2	0.5	0.2	0.1	0.5	0.4	0.4	0.4	0.4
degrees	1.0	3.0	5.0	5.0	5.0	5.0	0.0	0.0	0.0	0.0
translate	0.3	0.25	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.1
scale	0.5	0.6	0.8	0.8	0.7	0.8	0.5	0.9	0.9	0.5
shear	0.5	3.0	2.0	5.0	5.0	5.0	0.0	0.0	0.0	0.0
perspective	0.0	0.0008	0.0005	0.001	0.001	0.001	0.0	0.0	0.0	0.0
flipud	0.05	0.2	0.2	0.3	0.3	0.3	0.0	0.0	0.0	0.0
fliplr	0.1	0.4	0.2	0.6	0.3	0.5	0.5	0.5	0.5	0.5
mosaic	1.0	1.0	0.8	0.9	0.8	1.0	1.0	1.0	1.0	1.0
mixup	0.01	0.1	0.2	0.2	0.2	0.2	0.01	0.15	0.15	0.05
copy_paste	0.0	0.05	0.1	0.1	0.1	0.1	0.0	0.0	0.0	0.0
paste_in	0.2	0.0	0.2	0.1	0.1	0.1	0.2	0.15	0.15	0.05
loss_ota	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE 5.1: Hyperparameters of the experiments

We evaluated our proposed method on the test set and reported the performance using mean average precision (map) with different intersection over union (IoU) thresholds. Specifically, we report results for IoU thresholds of 0.5 and 0.95, which are commonly used in object detection tasks. Additionally, we report recall and precision to provide a more comprehensive understanding of the performance.

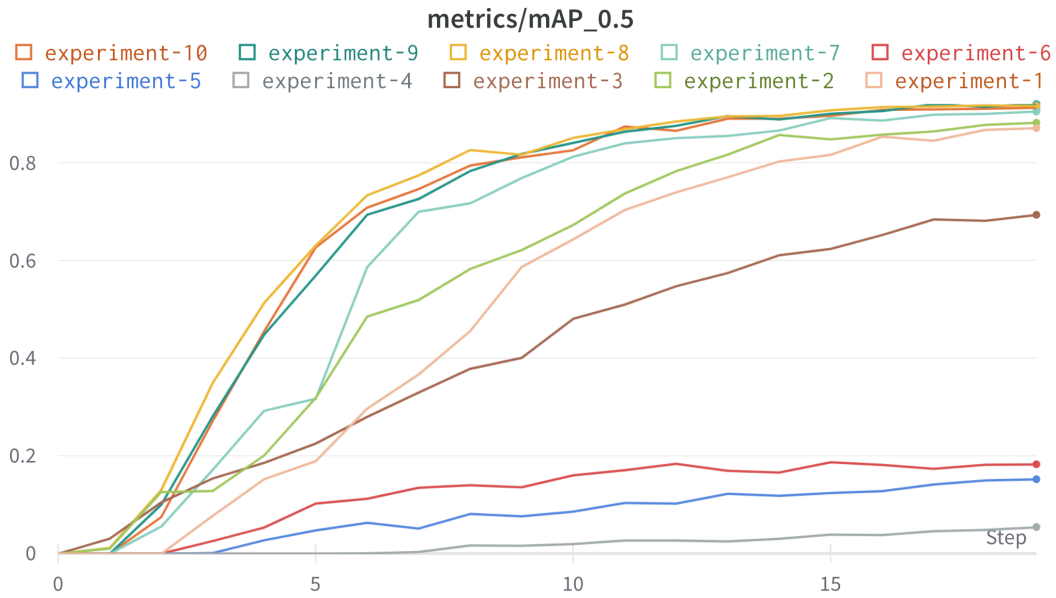

FIGURE 5.1: Mean Average Precision at IoU threshold of 0.5

First, let us look at the results for map@0.5. We tested the model with different hyperparameters and found that the best performance was achieved in Experiment 9, Table 5.1. Figure 5.1 shows that the model achieved a map@0.5 score of 0.9207.

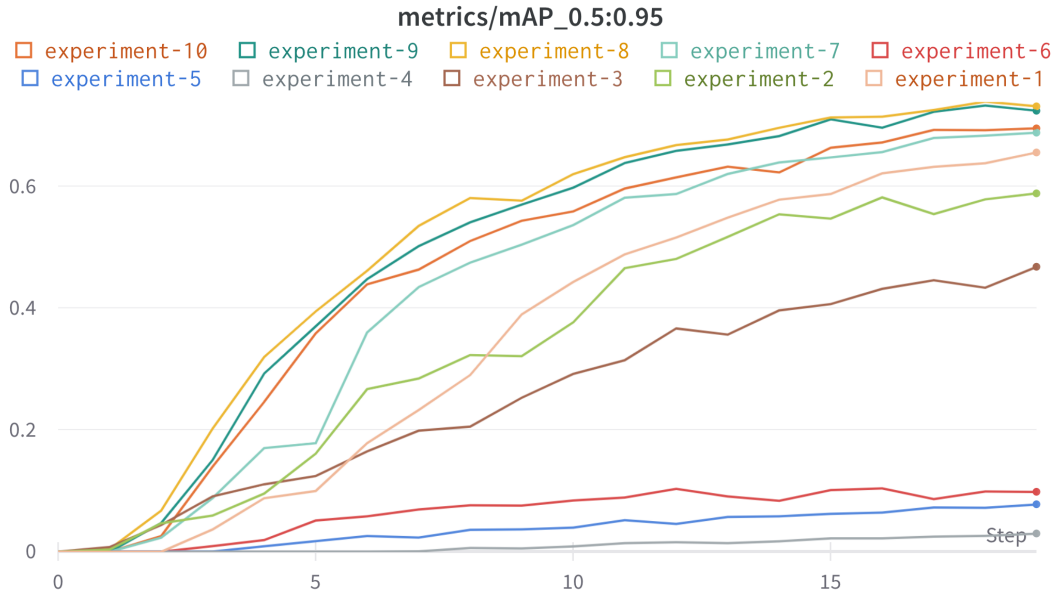

FIGURE 5.2: Mean Average Precision at IoU threshold of 0.95

Moving on to map@0.95, we observed similar performance trends, with the best hyperparameters in Experiment 8, Table 5.1. Figure 5.2 shows that the model achieved a map@0.95 score of 0.7308.

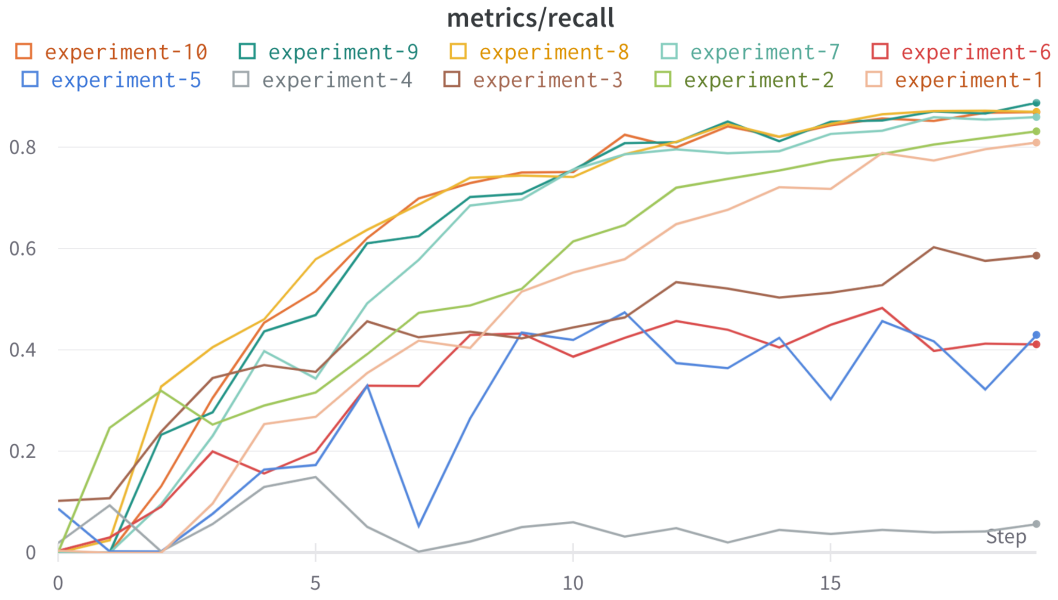

FIGURE 5.3: Recall results

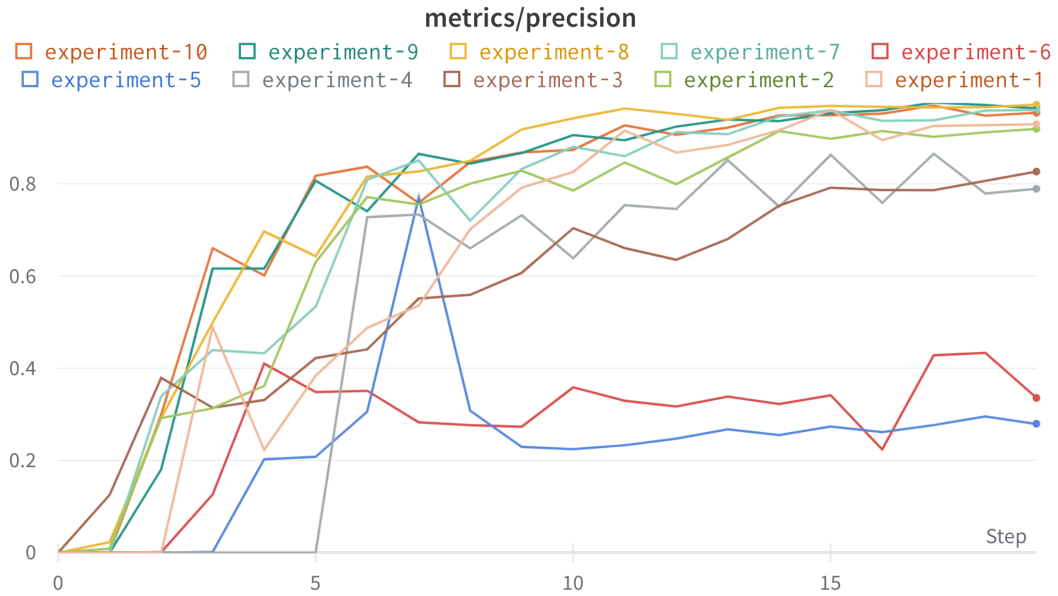

FIGURE 5.4: Precision results

Based on our experiments, we found that the best recall was achieved in Experiment 9, Table 5.1, as shown in Figure 5.3, with a value of 0.8878, while the best precision was achieved in Experiment 8, Table 5.1, as shown in Figure 5.4, with a value of 0.9719.

Our proposed method achieved competitive performance on the test set, indicating its effectiveness in addressing the problem. Furthermore, the achieved recall

and precision values demonstrate the model's ability to accurately detect the relevant objects in the images and reduce false positives.

5.1 Selection of the best hyperparameters

After evaluating different hyperparameters, we found that Experiment 9 achieved the best recall with a score of 0.8878 and the best map@0.5 with a score of 0.9207. Experiment 8 achieved the best precision with a score of 0.9719 and the best map0.95 with a score of 0.7308. Since Experiment 8 and Experiment 9 showed similar results, we took the hyperparameters from Experiment 8 for further research.

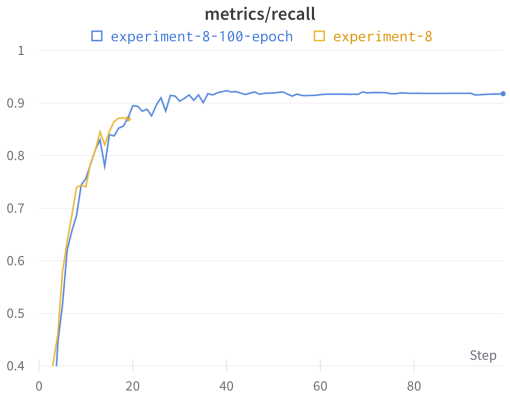

FIGURE 5.5: Recall results for 100 epoch

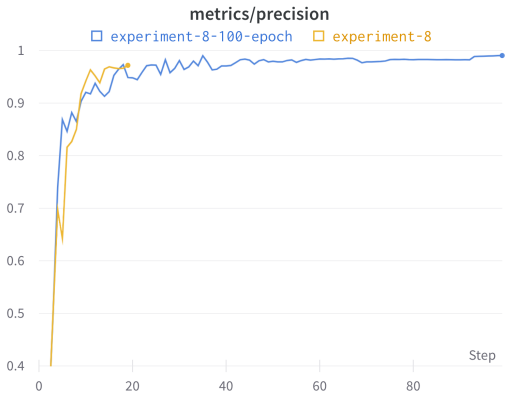

FIGURE 5.6: Precision results for 100 epoch

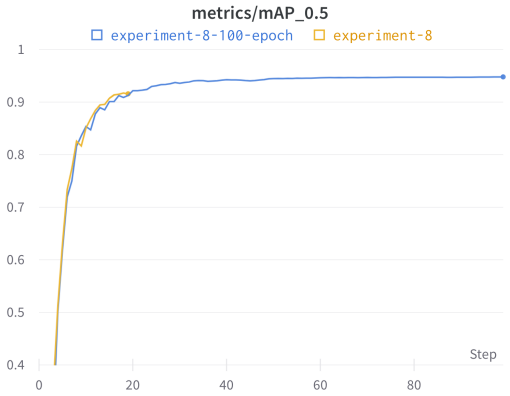

FIGURE 5.7: Mean Average Precision at IoU threshold of 0.5 for 100 epoch

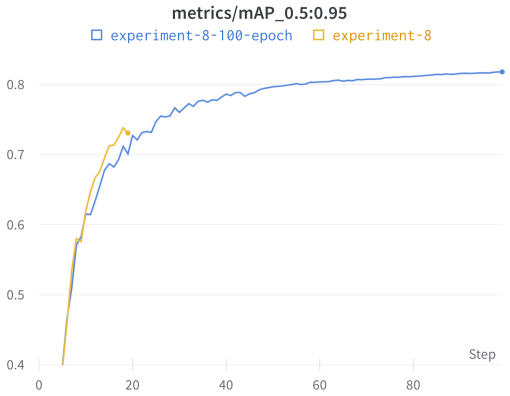

FIGURE 5.8: Mean Average Precision at IoU threshold of 0.95 for 100 epoch

To investigate the model's performance over time, we ran this configuration for 100 epochs and recorded each epoch's recall and precision scores. As shown in Figure 5.5, Figure 5.6, Figure 5.7 and Figure 5.8, the model's performance improved over time, with recall increasing from 0.87 to 0.9178, precision increasing from 0.9719 to 0.9905, map@0.5 increasing from 0.9158 to 0.948, map@0.95 increasing from 0.7308 to 0.8181 at the end.

These results suggest that the model can improve further with more training and that our proposed hyperparameters are effective for this task. We recommend using hyperparameters from Experiment 8 shown in Table 5.1.

5.2 Confusion matrix

A table known as a confusion matrix is frequently used to explain how well a classification model performs on a collection of data for which its true values are known. In our case, we have a 12-class classification problem where 11 classes represent different types of mines, and one class represents the false positive or true negative background.

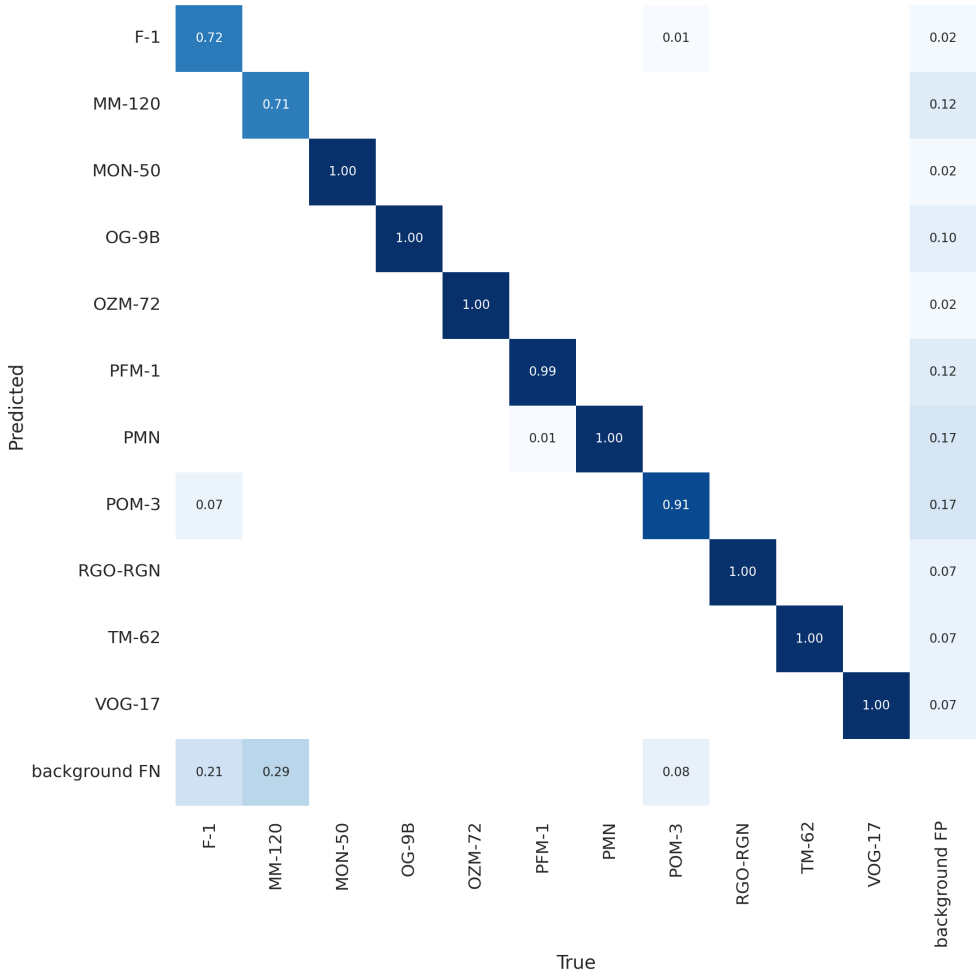

FIGURE 5.9: Confusion matrix

According to Figure 5.9, the model has high accuracy in detecting several types of mines and grenades, such as MON-50, OG-9B, OZM-72, PFM-1, PMN, POM-3, RGO/RGN, TM-62, and VOG-17, with true positive scores ranging from 0.91 to 1.00. However, the model struggles with the F-1 and MM-120 classes, with a true positive score of only 0.72 and 0.71, respectively.

On the false positive side, the model has a relatively low score for predicting background false positives for the mines and grenades, with scores ranging from 0.02 to 0.17.

In addition, the model has a clear score for predicting false negatives for some classes, including F-1, MM-120, and POM-3, with scores ranging from 0.08 to 0.29.

Overall, the model's performance is promising, with high accuracy in detecting most landmines and grenades. However, there is room for improvement in detecting the F-1 and MM-120 classes and reducing false positive and false negative predictions for some classes. Reducing false negative scores is of utmost importance to ensure the reliability and accuracy of the detection system. Improving the detection system's ability to identify and classify explosive materials and devices and minimizing the risk of false negatives can save lives, prevent damage, and enhance overall security measures. To improve, we plan to:

- Collect a larger dataset with more variability in mine instances;
- Augment the synthetic data with real ones;
- Fine-tune the model's hyperparameters and architecture.

5.3 Visualizations of the model's performance

FIGURE 5.10: Model predictions on the test set

We also provide visualizations of the model's performance on both the test and validation sets. As shown in Figure 5.10, the model could accurately detect and label objects in the test set. The detected objects are highlighted in different colors depending on the class. We can see that the model correctly identified all the objects in the image. Of course, according to the results, among the entire dataset, both false positive and true negative images did not fall into this sample of images.

FIGURE 5.11: Model predictions of the validation set

Similarly, in Figure 5.11, we show the model's performance on the validation set. Again, we can see that the model has accurately detected and labeled all the objects in the image.

Chapter 6

Project production

Our objective was to develop an effective solution to detect mines using YOLOv7 for further use in ground-based remote operations to detect mines and artillery shells.

To achieve this, we connected our trained weights to the Nvidia Jetson Xavier microcomputer, which allowed us to process the input images and detect mines in real time. The Jetson Xavier is a powerful microcomputer that has the capability to run complex algorithms and process large amounts of data quickly, making it an ideal choice for our project.

To launch the scales and begin the detection process, we utilized the `detect.py` method, which is part of the YOLOv7 model. This method is well-suited for object detection tasks and provides accurate and reliable results for our project.

Figure 6.1 visually represents the model that runs through a microcomputer using an IMX219-83 Stereo Camera. The photo was taken using one of the camera lenses.

FIGURE 6.1: Screenshot of the model running on the Nvidia Jetson Xavier microcomputer

After the detection process is completed, remote robots are deployed in the mine detection zone. The robots are equipped with tools to detect mines on the surface, which our model does, and underground, using a point metal detector. It was done in order to save the lives of sappers and remove them from the process of detecting mines for demining.

Chapter 7

Conclusions

This research aimed to develop a deep learning model for mine detection in remote demining operations. We trained and evaluated the model using a dataset of synthetic and real images, achieving high accuracy on both the validation and test sets for most classes. Our proposed method achieved competitive performance with the state-of-the-art models, showing promise for real-world deployment.

7.1 Result summary

Throughout this thesis, we have proposed a mine detection system that utilizes a state-of-the-art object detection model. We used the YOLOv7 architecture to train our model on a custom dataset of mine images. We evaluated our model on a test set and achieved a mAP@0.5 score of 0.948, indicating the high performance of our proposed method.

Finally, we installed our model with its weights onto the Nvidia Jetson Xavier for use in real-time on remote demining robots.

7.2 Future work

We plan to deploy the Nvidia Jetson Xavier microcomputer on remote demining robots in future work. We will create software to use a webcam to capture images of the terrain and the detected mines and collect GPS information to create a map with markings of found mines. Additionally, we plan to collect a larger dataset of images to replace synthetic data with real ones, adding variability of mine instances and generally increasing their number to improve the model's robustness.

Bibliography

[1] project 1. *metal Dataset*. Open Source Dataset. 2023. URL: https://universe.roboflow.com/project-1-quqvg/metal-u7dws.

[2] Apremeyan. *Garbage*. 2019. URL: https://www.kaggle.com/datasets/apremeyan/garbage.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[4] Y. Bochkovskiy, C. Wang, and H. Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *arXiv preprint arXiv:2106.09797* (2021). URL: https://arxiv.org/abs/2106.09797.

[5] H. Cai, L. Zhu, and S. Han. "ProxylessNAS: Direct neural architecture search on target task and hardware". In: *arXiv preprint arXiv:1812.00332* (2018). URL: https://arxiv.org/abs/1812.00332.

[6] cdsaml. *Metal Dataset*. Open Source Dataset. 2022. URL: https://universe.roboflow.com/cdsaml-9uriy/metal-oajyy.

[7] Jifeng Dai et al. "R-FCN: Object Detection via Region-based Fully Convolutional Networks". In: *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 379–388. URL: https://papers.nips.cc/paper_files/paper/2016/hash/577ef1154f3240ad5b9b413aa7346a1e-Abstract.html.

[8] T. Diwan. "Object detection using YOLO: challenges, architectural successors, datasets and applications". In: *International Journal of Advanced Computer Science and Applications* 11.7 (2020), pp. 23–27. URL: https://link.springer.com/article/10.1007/s11042-022-13644-y.

[9] Nazar Dobrovolskyy. *Mine detector*. 2023. URL: https://github.com/OutJeck/mine-detector.

[10] Juan Du. "Understanding of Object Detection Based on CNN Family and YOLO". In: *Journal of Physics: Conference Series* 1510.1 (2020), p. 012042. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1004/1/012029/meta.

[11] Brad Dwyer, Joseph Nelson, Joseph Solawetz, et al. *Roboflow (Version 1.0)*. Version 1.0. Computer vision software. 2022. URL: https://roboflow.com.

[12] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge." In: *Int. J. Comput. Vis.* 88.2 (2010), pp. 303–338. URL: http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10.

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[14] Kaiming He et al. "Mask R-CNN". In: *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. URL: https://arxiv.org/abs/1703.06870.

[15] Data Cluster Labs. *Domestic Trash - Garbage Dataset*. 2021. URL: https://www.kaggle.com/datasets/dataclusterlabs/domestic-trash-garbage-dataset.

[16] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2980–2988. URL: https://arxiv.org/abs/1708.02002.

[17] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list. 2014. URL: http://arxiv.org/abs/1405.0312.

[18] K. Liu et al. "Underwater target detection based on improved YOLOv7". In: (2023). URL: https://arxiv.org/pdf/2302.06939.pdf.

[19] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *European Conference on Computer Vision*. Springer. 2016, pp. 21–37. URL: https://arxiv.org/abs/1512.02325.

[20] V. Pham, D. Nguyen, and C. Donan. "Road Damages Detection and Classification with YOLOv7". In: (2022). URL: https://arxiv.org/abs/2211.00091.

[21] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. cite arxiv:1804.02767Comment: Tech Report. 2018. URL: http://arxiv.org/abs/1804.02767.

[22] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. URL: https://arxiv.org/abs/1506.02640.

[23] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. URL: https://arxiv.org/abs/1506.01497.

[24] Boris Sekachev et al. *opencv/cvat: v1.1.0*. Version v1.1.0. Aug. 2020. DOI: 10.5281/zenodo.4009388. URL: https://doi.org/10.5281/zenodo.4009388.

[25] C. Tan, R. Pang, and Q. V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 6105–6114. URL: https://arxiv.org/abs/1905.11946.

[26] Gary Thung. *trashnet*. Open Source Dataset. URL: https://github.com/garythung/trashnet/blob/master/data/dataset-resized.zip.

[27] Z. Tian et al. "FCOS: Fully Convolutional One-Stage Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9627–9636. URL: https://arxiv.org/abs/1904.01355.

[28] *Up to 30 percent of Ukrainian territory might be contaminated by mines and unexploded ordnance*. 2023. URL: https://war.ukraine.ua/war-news/up-to-30-percent-of-ukrainian-territory-might-be-contaminated-by-mines-and-unexploded-ordnance/.

[29] Zi Ang Wang. *Waste Pictures*. Online. 2021. URL: https://www.kaggle.com/datasets/wangziang/waste-pictures.