**Bharat Electronics Limited–Central Research Laboratory (BEL-CRL)**

# Landmine Detection using YOLOv8

**Intern:** S J Premkumar Nayaka

**Mentor:** Praveen jujjuruv

*Internship Project Report*

July 31, 2025

# Abstract

Unexploded landmines pose serious risks in defense and post-conflict zones, with tradi- tional detection methods being slow and dangerous. This project introduces an auto- mated landmine detection system using YOLOv8, a modern deep learning–based object detector capable of accurate, real-time performance. A custom dataset of diverse land- mine images was curated and used to fine-tune the model on CPU-only hardware, making it portable and easy to deploy in the field. The system can process aerial or ground-level images to detect landmines with high reliability, reducing the need for human interven- tion in hazardous environments. Its lightweight design also makes it ideal for integration with UAVs or ground robots, enabling large-area scanning and safer, faster operations.

**Keywords:** Landmine detection, Object detection, YOLOv8, Deep learning, Com- puter vision, Defense

# Contents

1. [Introduction](#_bookmark0) 3
2. [Dataset](#_bookmark1) 4
   1. [Source](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   2. [Classes](#_bookmark3) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   3. [Splits](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   4. [Input Handling](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
3. [Methodology](#_bookmark6) 6
   1. [YOLOv8 Overview](#_bookmark7) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   2. [Training Setup](#_bookmark8) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   3. [Implementation](#_bookmark10) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
4. [Results](#_bookmark11) 8
5. [Summary of Vitis AI User Guide](#_bookmark13) 10
6. [Conclusion](#_bookmark14) 12

**Chapter 1 Introduction**

Landmines remain a persistent threat in modern defense operations and post-conflict regions, posing severe risks to both civilians and military personnel. Traditional detection methods such as manual sweeping and metal detectors are slow, labor-intensive, and hazardous.

This project develops an automated landmine detection system using YOLOv8, a state-of-the-art deep learning–based object detection framework. YOLOv8 processes im- ages in a single pass, making it fast and suitable for real-time applications. By training on a custom dataset of landmine images captured under various terrains and lighting con- ditions, the model learns to accurately identify and localize landmines even in complex backgrounds.

The system eliminates the need for manual sweeping and metal detectors, significantly reducing the risk to personnel in mine-affected areas. Its lightweight design allows it to run efficiently on standard CPU hardware, making it highly portable and practical for field deployment. Furthermore, it can easily be integrated with drones or ground robots for large-area scanning, ensuring safer and more efficient landmine detection operations.

# Chapter 2 Dataset

## Source

The dataset used for this project was sourced from Roboflow Universe, an open repository for computer vision datasets. It is specifically curated for the landmine detection task and contains annotated images of landmines captured under various terrains and lighting conditions. The dataset follows the YOLO format, where each image has a corresponding label file containing bounding box coordinates and class information. It is divided into training, validation, and test sets to ensure reliable model evaluation. The images vary in resolution and background complexity, helping the model learn to generalize across different scenarios. This diversity in the dataset allows the YOLOv8 model to be robust against environmental challenges such as cluttered backgrounds, varying soil textures, and partial occlusions—key factors in real-world landmine detection tasks.

## Classes

This work focuses on a primary class:

* + - **landmine**

## Splits

Table 2.1: Dataset splits with number of images and annotations.

|  |  |  |
| --- | --- | --- |
| **Split** | **Images** | **Annotations** |
| Train | 3981 | 3981 |
| Validation | 380 | 380 |
| Test | 189 | 189 |

Landmine Detection using YOLOv8 BEL CRL Internship

## Input Handling

The inference pipeline accepts three types of inputs: a single image file, a video file, or a live webcam stream, specified through the INPUT\_SOURCE variable. If the input is an image path (.jpg,.png), the script performs single-image inference and saves the annotated image under runs/detect/predict/. If the input is a video path (m˙ p4), the script reads frames sequentially, applies preprocessing on each frame, performs detection, and writes the annotated video to the same output folder. If INPUT\_SOURCE is set to 0, the default system webcam is opened and processed in real time.

File-type detection is carried out using predefined extension sets (IMG\_EXTS and VID\_EXTS), and existence checks are performed to catch invalid paths early. For web- cams, the system validates whether the device is accessible, raising descriptive runtime errors on failure.

Before inference, the input undergoes a preprocessing pipeline designed to improve robustness under outdoor conditions such as shadows, dust, and variable lighting. The pipeline includes bilateral filtering for noise suppression, local contrast enhancement via CLAHE in Lab color space, gamma correction using an auto-gamma heuristic based on image brightness, and an optional letterbox resize to maintain aspect ratio. For videos, letterboxing is disabled by default to preserve the original frame size required by the video writer. The pipeline retains the BGR color format (OpenCV default) since Ultr- alytics’ YOLOv8 model handles color conversions internally and all annotated outputs (images or videos) are saved under runs/detect/predict/, consistent with Ultralytics’ de- fault directory structure. The folder is automatically created if it does not already exist, ensuring compatibility when initializing custom video writers and avoiding file-not-found errors during the save operation.

# Chapter 3 Methodology

## YOLOv8 Overview

YOLOv8 is a one-stage, anchor-free detector with decoupled heads for classification and regression. It predicts bounding boxes and class probabilities in a single forward pass, enabling real-time operation.

## Training Setup

Table [3.1](#_bookmark9) summarizes the training hyperparameters used for the YOLOv8n model. The network was trained for 15 epochs on images resized to 640 × 640 640×640 pixels, with a batch size of 4 to fit CPU-only constraints. The AdamW optimizer was employed to balance weight decay and convergence. The total loss combines bounding box regression, objectness, and classification components to optimize detection performance.

Table 3.1: Training hyperparameters.

**Parameter Value**

Model YOLOv8n (nano)

Epochs 15

Image size 640 *×* 640

Batch size 4

Optimizer AdamW

Losses Box + Objectness + Class

Landmine Detection using YOLOv8 BEL CRL Internship

## Implementation

The system was implemented in Python using the Ultralytics YOLOv8 framework (Py- Torch backend) for detection and OpenCV for image and video processing. All exper- iments were performed in a virtual environment on CPU-only hardware, with project outputs organized under Ultralytics’ default runs/ structure. The YOLO-format dataset was loaded through data.yaml, which defines the train, validation, and test splits along with class labels. Training used the YOLOv8n (nano) model with the hyperparame- ters in Table [3.1](#_bookmark9), saving the best-performing checkpoint as best.pt. Evaluation was carried out using Ultralytics’ model.val API on the held-out test split, while the in- ference script was designed to handle three input types: image files, video files, or live webcam streams, specified via the INPUT\_SOURCE parameter. Inputs are validated and type-checked through extension matching, and for videos/webcam, frames are processed sequentially with annotated videos saved using OpenCV’s VideoWriter. A lightweight preprocessing pipeline—comprising bilateral denoising, CLAHE-based local contrast en- hancement, and auto gamma correction—can be enabled to improve robustness under challenging lighting conditions; letterbox resizing is applied only for images to preserve output dimensions in videos. All annotated images and videos are automatically saved in runs/detect/predict/, ensuring consistency and preventing file-not-found errors when writing results.

# Chapter 4 Results

The trained YOLOv8n model was evaluated on the held-out test set comprising 189 images and 212 landmine instances. Table [4.1](#_bookmark12) summarizes the key detection metrics ob- tained during evaluation. The model achieved high precision and recall, indicating reli- able detection performance, and strong mean Average Precision (mAP) scores at standard thresholds.

Table 4.1: Detection metrics on the test set.

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Precision (mean) | 0.9675 |
| Recall (mean) | 0.9623 |
| F1 score (mean) | 0.9649 |
| [mAP@0.5](mailto:mAP@0.5) | 0.9908 |
| [mAP@0.75](mailto:mAP@0.75) | 0.7717 |
| mAP@0.5:0.95 | 0.6531 |
| Images evaluated | 189 |
| Instances | 212 |

The model attained a mean precision of **96.7%**, meaning nearly all predicted detec- tions were correct, and a mean recall of **96.2%**, indicating that the majority of actual landmines were successfully detected. The mean F1 score of **96.5%** reflects balanced per- formance between precision and recall. In terms of mean Average Precision (mAP), the model reached **[mAP@0.5](mailto:mAP@0.5) = 0.9908** at the standard IoU threshold, while **[mAP@0.75](mailto:mAP@0.75)**

**= 0.7717** and **mAP@0.5:0.95 = 0.6531** summarize its performance across stricter IoU thresholds.

Landmine Detection using YOLOv8 BEL CRL Internship

Figure 4.1: Sample detection with bounding box and confidence.



Figure 4.2: Another detection example in a different terrain.

# Chapter 5

**Summary of Vitis AI User Guide**

Vitis AI is Xilinx’s unified AI development platform for deploying machine learning work- loads on Xilinx hardware such as Versal ACAP, Zynq UltraScale+ MPSoC, and Alveo accelerator cards. It provides an end-to-end flow from model development to deployment and includes a set of optimized libraries, tools, and runtime environments.

## Vitis AI Tools

Vitis AI enables developers to take pre-trained deep learning models from popular frame- works (TensorFlow, PyTorch, Caffe, etc.) and optimize them for Xilinx hardware. The toolchain consists of:

* **Vitis AI Optimizer** – Prunes and quantizes models (e.g., 8-bit integer quantiza- tion) to reduce size and improve hardware efficiency without significant accuracy loss.
* **Vitis AI Compiler** – Compiles and maps the optimized model to a specific hard- ware target’s Deep Processing Unit (DPU).
* **Vitis AI Runtime (VART)** – Provides APIs for deploying compiled models on edge devices or data center cards.

## Supported Hardware and Architecture

The DPU is a configurable IP core optimized for deep learning inference, supporting convolutional neural networks (CNNs) and various layer types. Vitis AI supports edge devices (Zynq UltraScale+ MPSoC, Versal ACAP) and data center acceleration cards (Alveo). Deployment can be done through pre-built images or custom SD card images.

Landmine Detection using YOLOv8 BEL CRL Internship

## Model Flow

The typical workflow includes:

* + 1. Train or fine-tune a model using a popular deep learning framework.
    2. Quantize the model to INT8 using Vitis AI Quantizer.
    3. Compile the quantized model with the Vitis AI Compiler for a specific target DPU.
    4. Deploy the model with Vitis AI Runtime (VART) on the edge or data center device.

## Key Features

* + - * Supports popular deep learning frameworks (TensorFlow, PyTorch, Caffe).
      * Provides optimized libraries (Vitis AI Library) for common AI tasks like image classification, detection, and segmentation.
      * Enables hybrid programming (AI + traditional software) via integration with Vitis Unified Software Platform.
      * Offers evaluation kits, pre-compiled models (Model Zoo), and example applications to speed up development.

## Deployment

Vitis AI supports cloud-to-edge workflows. Pre-compiled models can be downloaded from the Vitis AI Model Zoo, or users can compile custom models. Applications use the Vitis AI Runtime to load compiled models and execute inference efficiently on the DPU hardware.

# Chapter 6 Conclusion

This project successfully developed an automated landmine detection system using the YOLOv8 object detection architecture. A custom dataset containing annotated landmine images was used to train the model, which achieved high detection performance with a mean precision of 96.7%, mean recall of 96.2%, and [mAP@0.5](mailto:mAP@0.5) of 99.1% on the held-out test set. The unified inference pipeline supports image, video, and live webcam inputs, making it versatile for field deployment. The results demonstrate that the system can accurately detect landmines across diverse scenes with minimal false positives and false negatives. This makes it suitable for integration with UAVs or ground robots to reduce human involvement in hazardous areas. The project establishes a strong foundation for deploying deep learning-based solutions in defense applications for safer and faster landmine detection.