|  |  |
| --- | --- |
| **ITI121 Assignment 2: Custom Object Detection Report**  **Fine-Tuning a yolo11s Model for Sea Turtle and Water Plant Detection** | |
| **Student Name:** | **Kee Eng Zhen** |
| **Student ID:** | **7120163A** |
| **Module:** | **ITI121-2025S2** |

**1. Introduction**

This project presents the development, fine-tuning, and deployment of a custom object detection system using a YOLO-based deep learning model to detect two non-COCO object classes: Sea Turtle and Water Plant. The objective was to adapt a pre-trained YOLO model to a domain-specific dataset and achieve high detection accuracy through systematic experimentation and optimization.

The implementation was carried out in Python using the Ultralytics YOLO framework, with training, validation, and inference fully automated via a Jupyter notebook. Model performance was evaluated using standard object detection metrics, including Precision, Recall, mAP@0.5, and mAP@0.5–0.95. All experiments were tracked using Weights & Biases (wandb), and the final optimized model was exported to ONNX format and deployed with a Gradio-based web interface on Hugging Face.

**2. Data Collection and Annotation**

**2.1 Dataset Collection and Statistics**

To meet the assignment requirement of using **non-COCO classes**, two custom categories were selected: **Sea Turtle** and **Water Plant**. Images were collected from publicly available sources such as Google Images, ensuring diversity in:

* Background environments
* Lighting conditions
* Object scale and orientation
* Partial occlusions

The final curated dataset consisted of:

| **Class** | **Number of Images** |
| --- | --- |
| Sea Turtle | 76 |
| Water Plant | 68 |
| **Total** | **144** |

A subset of **93 unique images** was used for validation and testing to ensure unbiased performance evaluation.

**2.2 Annotation Methodology and Dataset Split**

All images were manually annotated using **Roboflow**, where bounding boxes were drawn tightly around target objects to minimize background noise. The dataset was exported in **YOLO format**, which includes normalized bounding box coordinates and class labels.

The dataset was split as follows:

* **Training set:** 70%
* **Validation set:** 20%
* **Test set:** 10%

This split ensures sufficient data for model learning while preserving independent samples for evaluation.

**Dataset Links:**

* Roboflow Project:

<https://app.roboflow.com/keeluckykee/turtle1/6>

**3. Model Architecture and Training Process**

**3.1 Base Model Selection**

A **YOLO Small variant** (YOLOv11s / YOLOv8-style architecture as implemented in Ultralytics) was selected due to its balance between:

* Detection accuracy
* Training efficiency
* Suitability for real-time inference

The model was initialized with **pre-trained COCO weights**, enabling transfer learning and faster convergence on the custom dataset.

**3.2 Training Configuration and Loss Functions**

Training was conducted using the Ultralytics YOLO training pipeline, which optimizes a **multi-component loss function**, including:

* **Binary Cross-Entropy (BCE) Loss:**  
  Used for object classification
* **Distribution Focal Loss (DFL):**  
  Improves bounding box regression precision
* **CIoU Loss:**  
  Enhances localization accuracy by penalizing poor overlap and center distance

Key training parameters reflected in the Python notebook include:

* Custom number of epochs (30, 50, and 100)
* Transfer learning from COCO weights
* Automatic mixed precision (AMP)
* Validation after each epoch

**3.3 Fine-Tuning Experiments**

To improve performance, two major fine-tuning experiments were conducted.

**Experiment 1: Learning Rate Optimization**

* **Change:** Initial learning rate reduced from 0.01 to 0.005
* **Rationale:**  
  A lower learning rate allows more stable gradient updates and prevents catastrophic forgetting of pre-trained features, especially with a relatively small custom dataset.

**Experiment 2: Input Resolution Enhancement**

* **Change:** Input image size increased to 640 × 640
* **Rationale:**  
  Higher resolution preserves fine-grained details, improving detection of smaller objects such as distant turtles and thin water plants.

Both experiments were implemented directly in the Python training code and tracked using wandb.

**4. Model Evaluation and Results**

**4.1 Evaluation Metrics**

Model performance was evaluated on the validation set using industry-standard metrics:

* **Precision (P@0.5)**
* **Recall (R@0.5)**
* **mAP@0.5**
* **mAP@0.5–0.95**

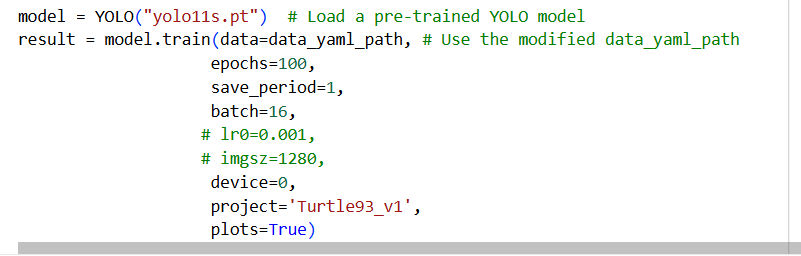
These metrics were automatically computed by the YOLO validation pipeline during training.

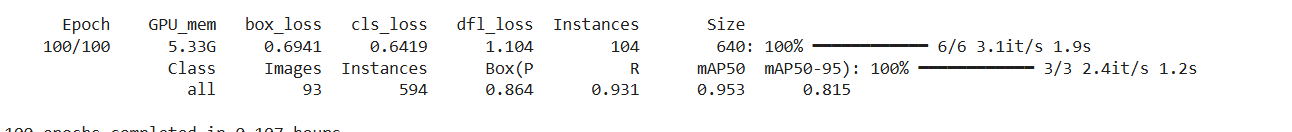
**4.2 Final Model Performance**

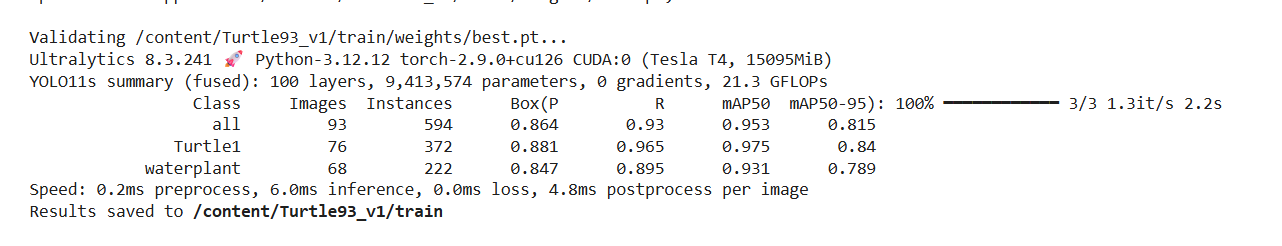
The final optimized model achieved the following results:

| **Metric** | **All Classes** | **Sea Turtle** | **Water Plant** |
| --- | --- | --- | --- |
| Precision | 0.864 | 0.881 | 0.847 |
| Recall | 0.930 | 0.965 | 0.895 |
| [mAP@0.5](mailto:mAP@0.5) | 0.953 | 0.975 | 0.931 |
| mAP@0.5–0.95 | 0.815 | 0.840 | 0 .789 |

The results demonstrate strong generalization and balanced detection performance across both object classes, with particularly high recall for sea turtles.







**5. Experiment Tracking and Deployment**

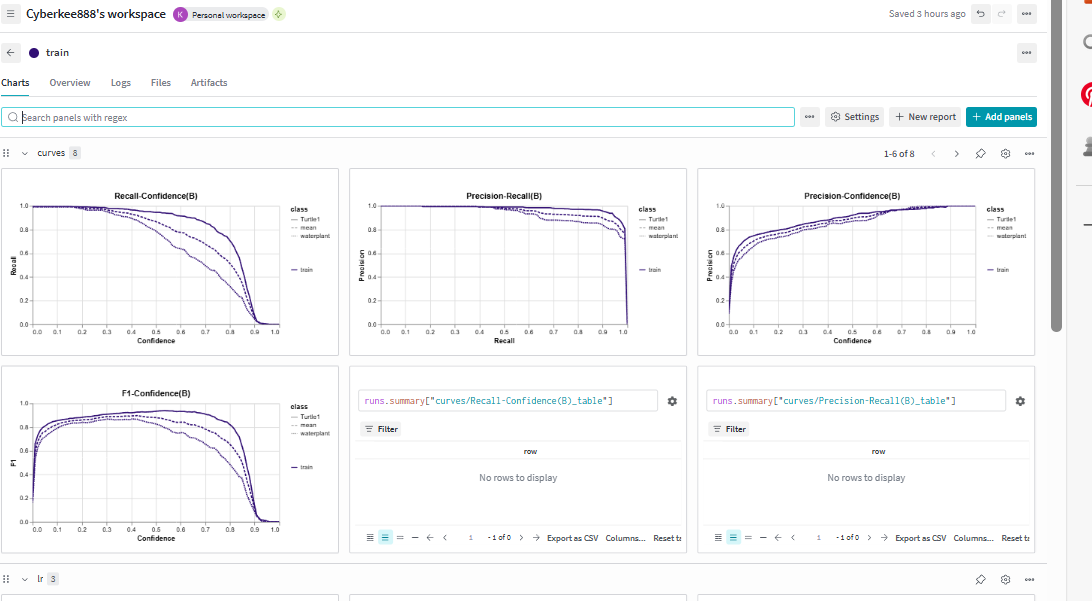
**5.1 Weights & Biases (wandb)**

All training runs were logged using **Weights & Biases**, enabling:

* Visualization of loss curves
* Comparison of hyperparameter configurations
* Monitoring of validation metrics across epochs

**wandb Run Link:**  
<https://wandb.ai/keeluckykee-nanyang-polytechnic/Turtle93_v1/runs/4vl2wzvs>

**[Placeholder for Screenshot of your experimental log on Weights & Biases showing the comparison between runs]**

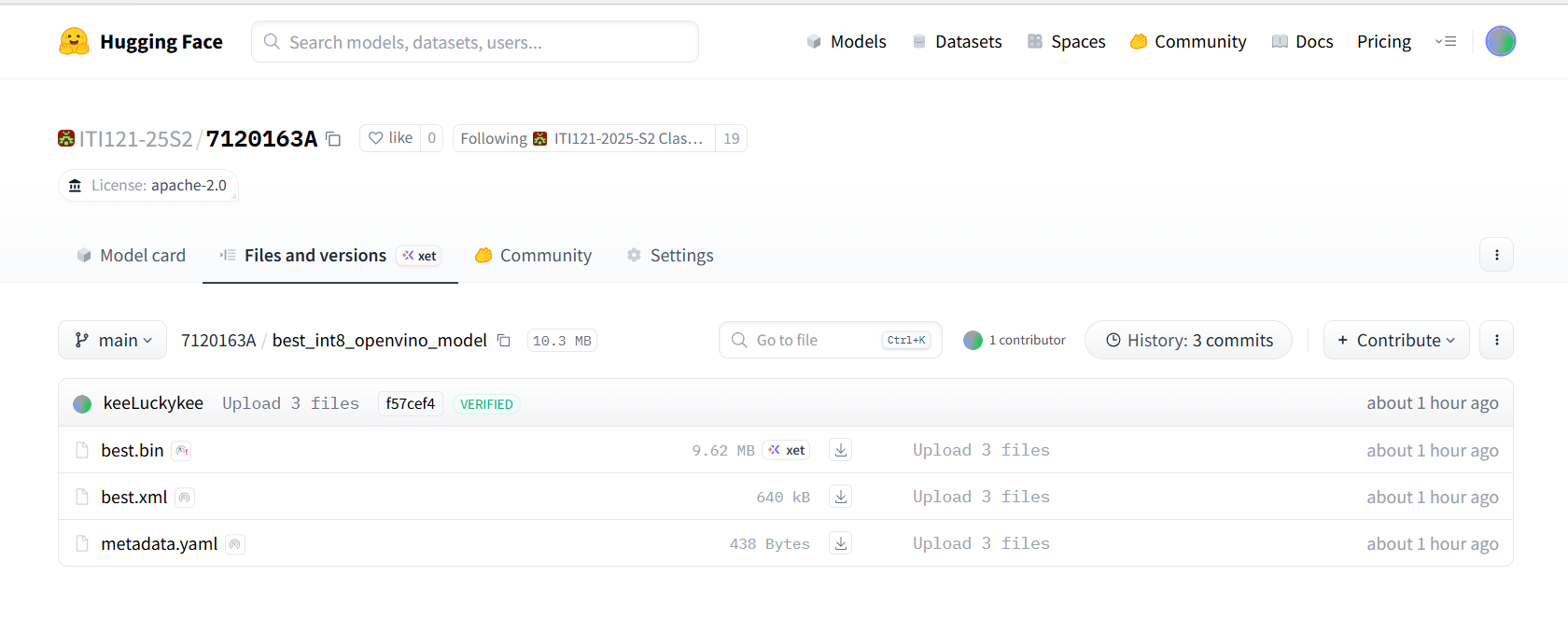


**5.2 Model Export and Deployment**

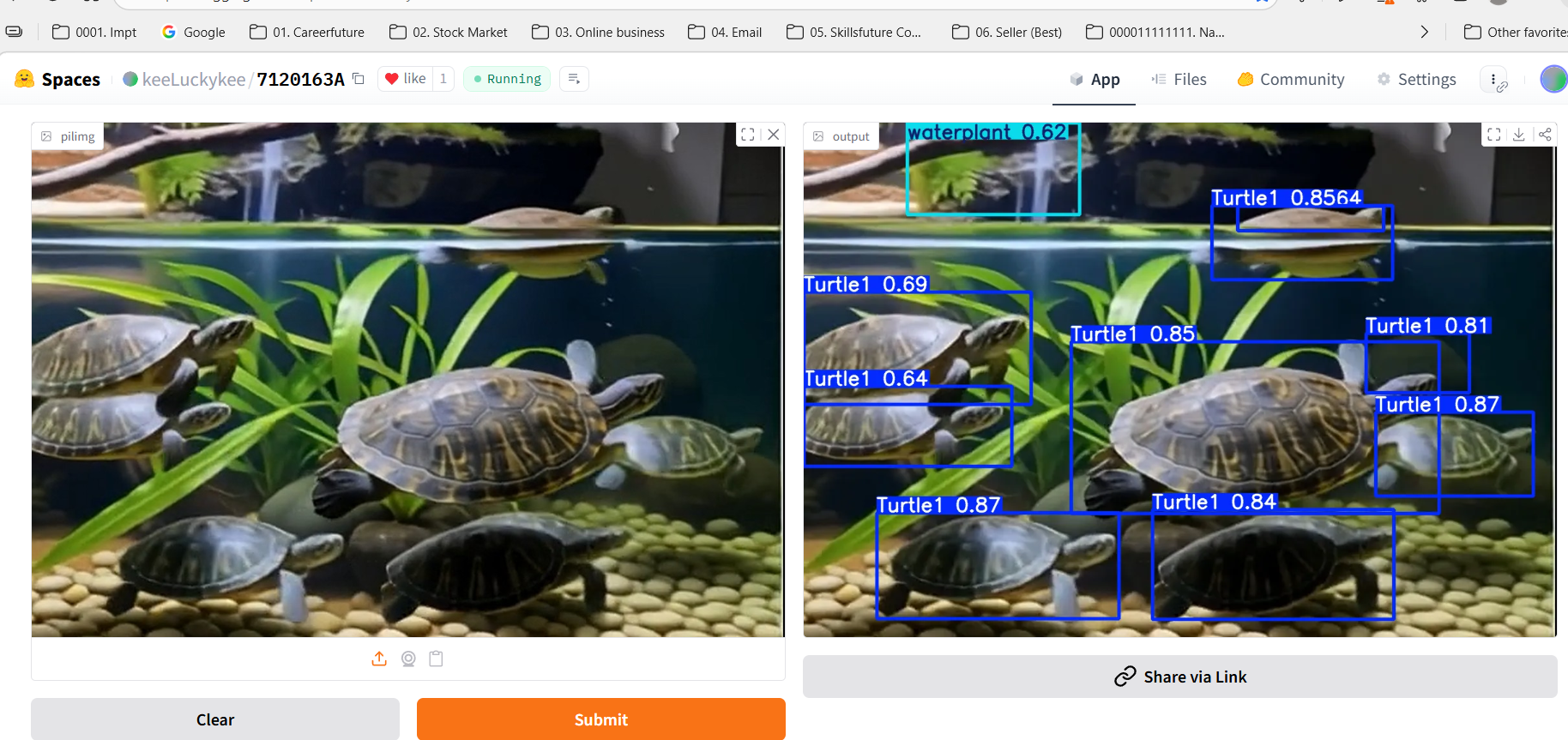
The final trained model was exported to **ONNX format**, enabling efficient CPU-based inference and cross-platform deployment.

A **Gradio web application** was built and deployed on **Hugging Face Spaces**, allowing users to upload images or videos and view real-time detection results.

* Hugging Face Model Repository:  
  *ITI121-2025S2/7120163A/* *best\_int8\_openvino\_model*



* Hugging Face Gradio App: *ITI121-2025S2/7120163A/*



GitHub: [Cyberkee123/ObjectDetectionTurtlewaterplantIT1121Assignment](https://github.com/Cyberkee123/ObjectDetectionTurtlewaterplantIT1121Assignment)

https://github.com/Cyberkee123/ObjectDetectionTurtlewaterplantIT1121Assignment

**6. Conclusion**

This project successfully demonstrates the end-to-end development of a **custom object detection system** using YOLO and transfer learning. Through careful dataset preparation, targeted fine-tuning (learning rate and input resolution), and systematic experiment tracking, the final model achieved a high **mAP@0.5 of 0.953**.

The project fulfills all assignment requirements, showcasing practical skills in:

* Dataset creation and annotation
* Deep learning model fine-tuning
* Performance evaluation
* Experiment logging
* Real-world model deployment

Sample predicted images and videos are attached to illustrate inference results.

Attach is the Predicted image and video.\



