National College of Ireland

# Configuration Manual

MSc Research Project
AI For Business


## Mohit Gummaraj Kishore
Student ID: x23309326

School of Computing
National College of Ireland


Supervisor: Charlyn Rosales

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Mohit Gummaraj Kishore |
| **Student ID:** | x23309326 |
| **Programme:** | AI For Business |
| **Year:** | 2025 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Charlyn Rosales |
| **Submission Due Date:** | 12/12/2025 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 650 |
| **Page Count:** | 4 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | |
| **Date:** | 28th January 2026 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual

Mohit Gummaraj Kishore
x23309326

## 1 Requirements

Running the notebook requires a set of accounts, software tools, and hardware resources. A valid Google account is necessary to access Google Colab, which serves as the primary execution environment for this project. The user must also have an active Kaggle account, since the dataset is retrieved through authenticated Kaggle API access. Kaggle credentials are stored in the `kaggle.json` file, which the notebook expects during the data loading stage. These two accounts form the minimum identity requirements needed to reproduce the experiments.

From a software perspective, Google Colab provides an online environment with Python, PyTorch, and the Ultralytics YOLOv8 framework already installed or available through package managers. The notebook relies on Colab's ability to install additional dependencies as required. No local installations are needed, and all operations are performed within the hosted Jupyter environment. The integration with Google Drive allows the user to store checkpoints, export results, and preserve experiment logs.

To meet performance expectations, the user must enable access to a GPU within Google Colab. The Tesla T4 GPU runtime supports the training and inference speed required for YOLOv8n experiments. CPU-only execution is possible but significantly slower and not recommended for full training. The hardware and software combination outlined above ensures that the notebook runs in a stable and reproducible manner.

## 2 Steps to Run the Notebook

To execute the notebook in Google Colab, the user begins by uploading the `.ipynb` file to their Google Drive or by opening it directly from the Colab interface. Once loaded, the user must select the appropriate runtime settings by navigating to *Runtime* → *Change runtime type* and enabling GPU acceleration. Google Colab provides access to the Tesla T4 GPU, which improves the speed and reliability of training and inference operations. After confirming the runtime configuration, the user proceeds by running the notebook cells in sequential order using the *Run all* option or by executing each cell manually.

The notebook includes optional integration with Google Drive to store model weights and logs. The user may enable or disable this integration depending on their preference for persistent storage. For dataset access, the notebook requires the user to upload the `kaggle.json` file containing their API credentials. This file authorizes the download of the dataset through the Kaggle API. The notebook validates the presence of the credentials and proceeds with dataset extraction and preprocessing.

Once these steps are completed, the user follows the execution flow defined in the notebook. Each block of code contains descriptive output that confirms data loading,

model configuration, training progress, evaluation metrics, and result generation. The process ensures a structured and repeatable experiment workflow.

# 3 Running the Notebook

## 3.1 Accounts and Setup Requirements

To execute the notebook, users must have access to a valid Google Account to log into Google Colab. Additionally, a Kaggle account is required to obtain an API token (‘kaggle.json‘) for downloading datasets directly from Kaggle. The notebook runs entirely in a cloud environment and does not require any local setup. Users must select a GPU runtime to ensure reasonable training times. The preferred environment is Google Colab with a T4 GPU backend. This hardware setup ensures compatibility with Ultralytics’ YOLOv8 implementation and provides adequate memory for model training and evaluation.

## 3.2 Step-by-Step Instructions

### Step 1: Install Dependencies

```
!pip install ultralytics
```

Install the Ultralytics package that includes YOLOv8 functionality. Wait for the installation to complete before moving forward.

### Step 2: Import Libraries

```
from ultralytics import YOLO
import os
```

These libraries are essential for training and evaluating the model.

### Step 3: Mount Google Drive (Optional)

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounting Google Drive allows you to save model weights and outputs persistently. Follow the prompts to authorize access.

### Step 4: Upload Kaggle API Credentials

```
from google.colab import files
files.upload()
```

Upload your `kaggle.json` file, which contains your Kaggle API key.

**Step 5: Set Kaggle Environment and Download Dataset**

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d snehilsanyal/construction-site-safety-image-dataset-robo
!unzip construction-site-safety-image-dataset-roboflow.zip
```

These commands authenticate your Kaggle session, download the dataset, and extract the contents into the working directory.

**Step 6: Train YOLOv8n Model**

```
model = YOLO("yolov8n.pt")
model.train(data="Construction_Site_Safety_Image_Dataset/data.yaml", epochs=50)
```

Initialize the YOLOv8n model and begin training. Adjust the number of epochs (e.g., 25 or 75) based on the experiment being conducted.

**Step 7: Validate the Trained Model**

```
metrics = model.val()
```

Evaluate the model on the validation set. Key metrics such as mAP, precision, and recall are displayed.

**Step 8: Run Inference on Test Images**

```
results = model.predict(source="Construction_Site_Safety_Image_Dataset/test/images",
```

Run predictions on unseen test images. Output files will be saved to the `runs/predict` directory.

**Step 9: Plot Training Metrics**

```
model.plot_metrics()
```

Generate training curves that show trends for loss, precision, recall, and mAP. This step is useful for diagnosing overfitting or underfitting.

## 3.3 Expected Output

Once executed correctly, each cell produces output immediately beneath it. After training is complete, validation metrics will appear as tabulated logs, and predictions on test images will be saved with bounding boxes drawn. The training graphs will illustrate learning trends across the specified epochs. The final model can be exported or used for integration with the safety monitoring web application. The entire notebook provides a reproducible pipeline for training, evaluation, and inference using YOLOv8n on a construction safety dataset.

# 4 Expected Output

When the notebook is executed, each cell produces intermediate results that guide the user through the training and evaluation process. The notebook runs one cell after another in a structured pipeline that begins with dataset loading and continues through model configuration, training, validation, and inference. Output from each cell appears directly below it, allowing the user to observe training progress, verify data integrity, and confirm that the model is learning effectively. Typical output includes loss curves, mAP scores, precision and recall metrics, and log messages that document the status of each training epoch.

As the model trains, the notebook displays plots that illustrate the evolution of performance metrics. These visual outputs help the user evaluate convergence and identify which epoch delivers the best validation performance. At the end of execution, the notebook reports the final experiment results, including mAP@0.5, mAP@0.5:0.95, precision, recall, and the path to the saved model weights. The expected output also includes inference samples that demonstrate the model's ability to detect PPE items on unseen images. By completing all cells successfully, the user obtains a full set of training results, performance comparisons, and generated figures that document the behavior of the model throughout the experiment.

# References