L-Università
ta' Malta

Advanced Computer Vision for AI – Generative AI Journal

ARI3129

Students:

Matthias Vassallo Pulis: 0198504L
Jacob Zammit: 320004L
Kyle Zammit: 57201L

Table of Contents

Introduction

Throughout the whole project, ChatGPT was the only Generative AI application that was used. ChatGPT is a generative AI chatbot that was developed by OpenAI. This tool makes use of OpenAI’s large language models (LLMs), such as GPT-4o [1], which is the most recent model available. It can perform a wide range of different tasks, like having a simple conversation with the user, brainstorming ideas, and providing assistance for understanding or fixing certain bugs in already developed code.

This application was chosen not only due to its popularity, but also because of its efficiency at performing all the aforementioned tasks with incredible efficiency. This shows the incredible versatility that ChatGPT provides. This has enabled us to enhance our quality of work by providing key insights through the responses it gives, especially when it comes to reviewing code. This is despite the fact that there also exist other tools which are more fine-tuned to deal with this specific task, such as Claude and Microsoft Copilot.

Ethical Considerations

Whilst developing the project, it was made essential that all the ethical aspects of Generative AI were understood. This was very important given the task of creating and training the various domestic waste bag object detection models used. These aspects include data privacy concerns, as well as biases that may occur revolving on how to improve model accuracy.

While using any type of generative AI, it is very important to bear in mind that these tools are prone to produce biases that may favour one specific approach over another. These can be reduced by using diverse training data. During the implementation of the object detection models, a lot of importance was made to ensure all responses generated by ChatGPT were free from any sort of bias, thus not impacting our findings. It was equally important to make sure that AI-generated content was used to support our work, instead of replacing it, thus maintaining the integrity and authenticity of the project.

In some cases, especially when generative AI is being used for code enhancements, users might be asked to provide existing data or code. This can lead to privacy concerns, especially if there is any sensitive data, such as API keys or any confidential data. A lot of care was taken throughout the project to ensure that no such data was passed to the application, especially when developing the object detection models. This was not a major issue when building the dataset, since none of the captured images included any sensitive information.

Methodology

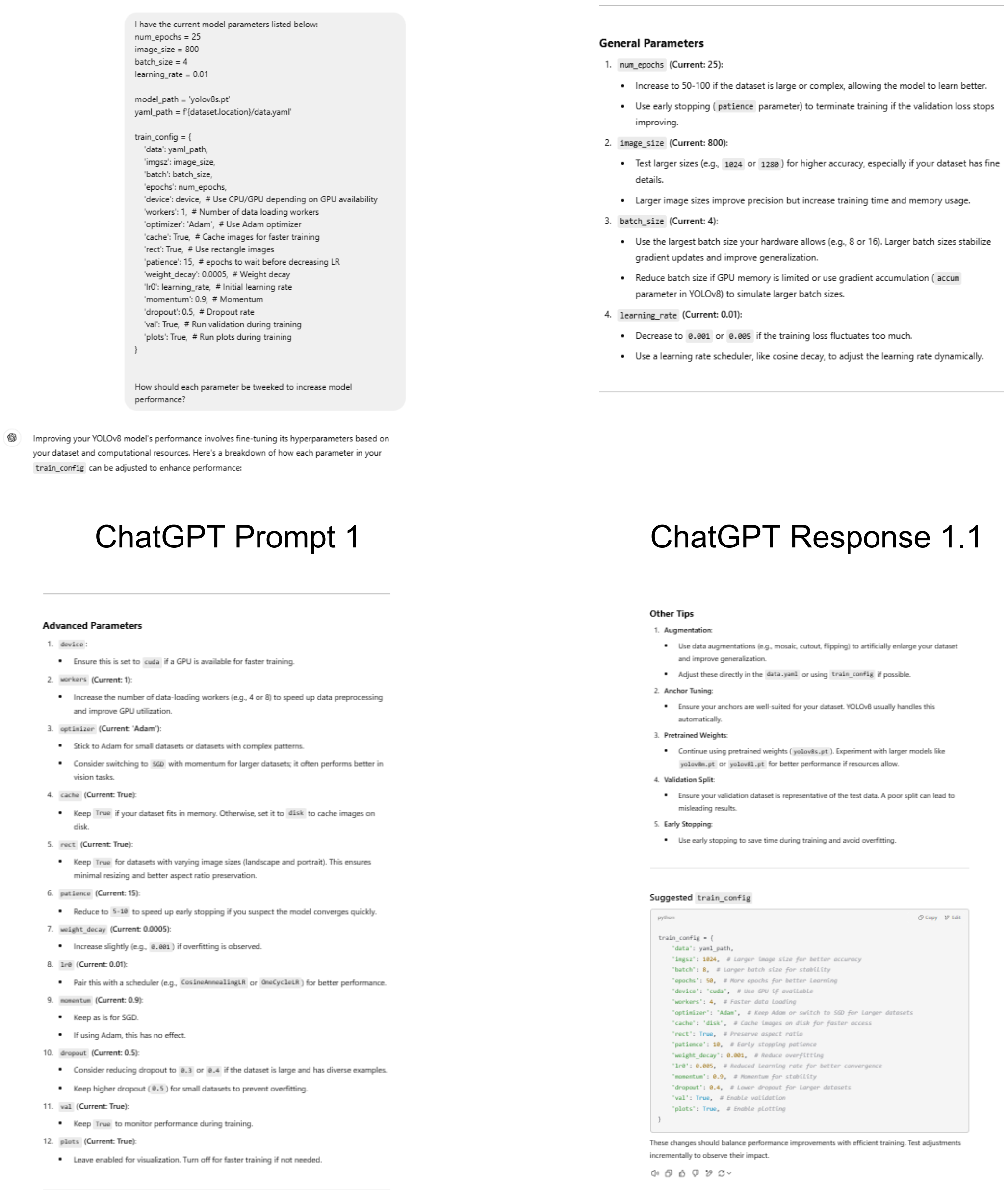
In order to make sure that generative AI was integrated into our project in a fair way, a number of key steps were made. Throughout the project, two approaches were utilised, one for developing the code, and another for writing the documentation.

During the development phase of the program, the code was first written and understood as much as possible. This was done until there was a point of difficulty, in which the official documentation provided by Ultralytics was consulted to try to look for any potential fixes. If no fix was found, or if there is a certain part of code that could not be understood, then ChatGPT was prompted accordingly. In the prompt, it was important to emphasise the type of response that would be given in order to maximise the efficiency of the generated responses. Once the response has been generated, it was manually reviewed carefully to ensure that there were no biases and that it is aligned with the needs of the task.

A similar process was done whilst writing the documentation. If there was a point in the writing which needed improvement, then the parts in question were copied and pasted into ChatGPT as prompts, making sure to ask it for suggestions on how they can be improved. Each prompt was double-checked to ensure that there were no privacy issues or any biases. Once the response was generated, it was again carefully reviewed to ensure that it suited what was required for the particular section. ChatGPT was only used in this part of the project for enhancing the grammar and structure, rather than producing informative content, as an extensive amount of research was carried out to ensure that the information was accurate and correct.

Prompts and Responses

Whilst experimenting with the hyperparameters for training the YOLOv8 model, there was a curiosity on how object detection performance can be improved by simply tuning them. Thus, ChatGPT was prompted to give suggestions on what hyperparameters to adjust and how they should be adjusted. This produced immediate results, with improved model accuracy, precision and recall. This can be seen in Figure 1.



```
I have the current model parameters listed below:
num_epochs = 25
image_size = 800
batch_size = 4
learning_rate = 0.01

model_path = 'yolov8s.pt'
yaml_path = f'{dataset.location}/data.yaml'

train_config = {
    'data': yaml_path,
    'imgsz': image_size,
    'batch': batch_size,
    'epochs': num_epochs,
    'device': device,  # Use CPU/GPU depending on GPU availability
    'workers': 1,  # Number of data loading workers
    'optimizer': 'Adam',  # Use Adam optimizer
    'cache': True,  # Cache images for faster training
    'rect': True,  # Use rectangle images
    'patience': 15,  # epochs to wait before decreasing LR
    'weight_decay': 0.0005,  # Weight decay
    'lr0': learning_rate,  # Initial learning rate
    'momentum': 0.9,  # Momentum
    'dropout': 0.5,  # Dropout rate
    'val': True,  # Run validation during training
    'plots': True,  # Run plots during training
}


How should each parameter be tweeked to increase model
performance?
```

Improving your YOLOv8 model's performance involves fine-tuning its hyperparameters based on your dataset and computational resources. Here's a breakdown of how each parameter in your `train_config` can be adjusted to enhance performance:

ChatGPT Prompt 1

General Parameters

1. `num_epochs` **(Current: 25):**
   - Increase to 50-100 if the dataset is large or complex, allowing the model to learn better.
   - Use early stopping (`patience` parameter) to terminate training if the validation loss stops improving.
2. `image_size` **(Current: 800):**
   - Test larger sizes (e.g., `1024` or `1280`) for higher accuracy, especially if your dataset has fine details.
   - Larger image sizes improve precision but increase training time and memory usage.
3. `batch_size` **(Current: 4):**
   - Use the largest batch size your hardware allows (e.g., 8 or 16). Larger batch sizes stabilize gradient updates and improve generalization.
   - Reduce batch size if GPU memory is limited or use gradient accumulation (`accum` parameter in YOLOv8) to simulate larger batch sizes.
4. `learning_rate` **(Current: 0.01):**
   - Decrease to `0.001` or `0.005` if the training loss fluctuates too much.
   - Use a learning rate scheduler, like cosine decay, to adjust the learning rate dynamically.

ChatGPT Response 1.1

Advanced Parameters

1. `device` :
   - Ensure this is set to `cuda` if a GPU is available for faster training.
2. `workers` **(Current: 1):**
   - Increase the number of data-loading workers (e.g., 4 or 8) to speed up data preprocessing and improve GPU utilization.
3. `optimizer` **(Current: 'Adam'):**
   - Stick to Adam for small datasets or datasets with complex patterns.
   - Consider switching to `SGD` with momentum for larger datasets; it often performs better in vision tasks.
4. `cache` **(Current: True):**
   - Keep `True` if your dataset fits in memory. Otherwise, set it to `disk` to cache images on disk.
5. `rect` **(Current: True):**
   - Keep `True` for datasets with varying image sizes (landscape and portrait). This ensures minimal resizing and better aspect ratio preservation.
6. `patience` **(Current: 15):**
   - Reduce to `5-10` to speed up early stopping if you suspect the model converges quickly.
7. `weight_decay` **(Current: 0.0005):**
   - Increase slightly (e.g., `0.001`) if overfitting is observed.
8. `lr0` **(Current: 0.01):**
   - Pair this with a scheduler (e.g., `CosineAnnealingLR` or `OneCycleLR`) for better performance.
9. `momentum` **(Current: 0.9):**
   - Keep as is for SGD.
   - If using Adam, this has no effect.
10. `dropout` **(Current: 0.5):**
    - Consider reducing dropout to `0.3` or `0.4` if the dataset is large and has diverse examples.
    - Keep higher dropout (`0.5`) for small datasets to prevent overfitting.
11. `val` **(Current: True):**
    - Keep `True` to monitor performance during training.
12. `plots` **(Current: True):**
    - Leave enabled for visualization. Turn off for faster training if not needed.

ChatGPT Response 1.2

Other Tips

1. **Augmentation:**
   - Use data augmentations (e.g., mosaic, cutout, flipping) to artificially enlarge your dataset and improve generalization.
   - Adjust these directly in the `data.yaml` or using `train_config` if possible.
2. **Anchor Tuning:**
   - Ensure your anchors are well-suited for your dataset. YOLOv8 usually handles this automatically.
3. **Pretrained Weights:**
   - Continue using pretrained weights (`yolov8s.pt`). Experiment with larger models like `yolov8m.pt` or `yolov8l.pt` for better performance if resources allow.
4. **Validation Split:**
   - Ensure your validation dataset is representative of the test data. A poor split can lead to misleading results.
5. **Early Stopping:**
   - Use early stopping to save time during training and avoid overfitting.

Suggested `train_config`

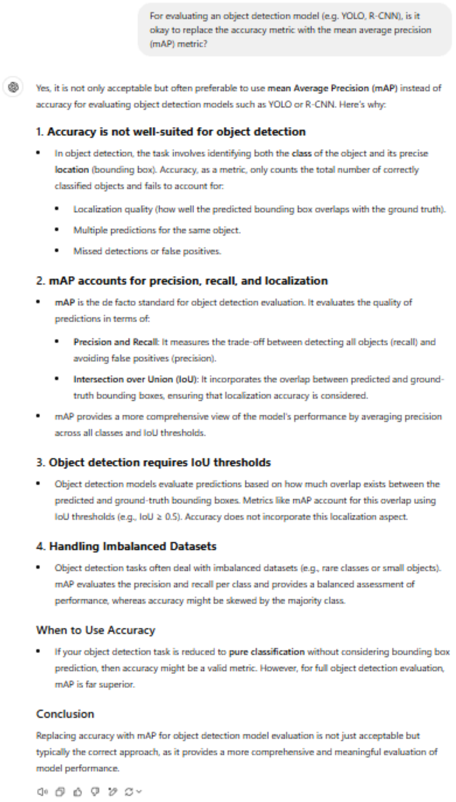
```python
train_config = {
    'data': yaml_path,
    'imgsz': 1024,  # Larger image size for better accuracy
    'batch': 8,  # Larger batch size for stability
    'epochs': 50,  # More epochs for better learning
    'device': 'cuda',  # Use GPU if available
    'workers': 4,  # Faster data loading
    'optimizer': 'Adam',  # Keep Adam or switch to SGD for larger datasets
    'cache': 'disk',  # Cache images on disk for faster access
    'rect': True,  # Preserve aspect ratio
    'patience': 10,  # Early stopping patience
    'weight_decay': 0.001,  # Reduce overfitting
    'lr0': 0.005,  # Reduced learning rate for better convergence
    'momentum': 0.9,  # Momentum for stability
    'dropout': 0.4,  # Lower dropout for larger datasets
    'val': True,  # Enable validation
    'plots': True,  # Enable plotting
}
```

These changes should balance performance improvements with efficient training. Test adjustments incrementally to observe their impact.

ChatGPT Response 1.3

Figure 1: ChatGPT Example Prompt 1

Throughout the evaluation phase, we encountered a situation where we didn’t know if we should primarily focus on the accuracy or the mean average precision (mAP), especially since model accuracy was not provided after training or validation. After querying ChatGPT, we came to the conclusion that mAP is more suited for this task, as can be seen in Figure 2. This gave us an idea on which metrics to give priority during evaluation.



For evaluating an object detection model (e.g. YOLO, R-CNN), is it okay to replace the accuracy metric with the mean average precision (mAP) metric?

Yes, it is not only acceptable but often preferable to use **mean Average Precision (mAP)** instead of accuracy for evaluating object detection models such as YOLO or R-CNN. Here's why:

1. Accuracy is not well-suited for object detection

- In object detection, the task involves identifying both the **class** of the object and its precise **location** (bounding box). Accuracy, as a metric, only counts the total number of correctly classified objects and fails to account for:
    - Localization quality (how well the predicted bounding box overlaps with the ground truth).
    - Multiple predictions for the same object.
    - Missed detections or false positives.

2. mAP accounts for precision, recall, and localization

- **mAP** is the de facto standard for object detection evaluation. It evaluates the quality of predictions in terms of:
    - **Precision and Recall**: It measures the trade-off between detecting all objects (recall) and avoiding false positives (precision).
    - **Intersection over Union (IoU)**: It incorporates the overlap between predicted and ground-truth bounding boxes, ensuring that localization accuracy is considered.
- mAP provides a more comprehensive view of the model's performance by averaging precision across all classes and IoU thresholds.

3. Object detection requires IoU thresholds

- Object detection models evaluate predictions based on how much overlap exists between the predicted and ground-truth bounding boxes. Metrics like mAP account for this overlap using IoU thresholds (e.g., IoU ≥ 0.5). Accuracy does not incorporate this localization aspect.

4. Handling Imbalanced Datasets

- Object detection tasks often deal with imbalanced datasets (e.g., rare classes or small objects). mAP evaluates the precision and recall per class and provides a balanced assessment of performance, whereas accuracy might be skewed by the majority class.

When to Use Accuracy

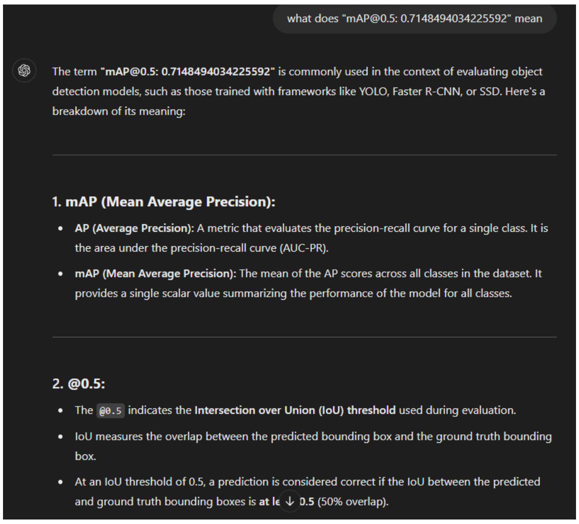
- If your object detection task is reduced to **pure classification** without considering bounding box prediction, then accuracy might be a valid metric. However, for full object detection evaluation, mAP is far superior.

Conclusion

Replacing accuracy with mAP for object detection model evaluation is not just acceptable but typically the correct approach, as it provides a more comprehensive and meaningful evaluation of model performance.

Figure 2: ChatGPT Example Prompt 2

Whilst experimenting and trying to interpret what the results of the training metrics of the YOLOv11 model mean, there was shorthand notation of ‘mAP@0.5’ and ‘mAP@0.50:0.95:’, which had never been encountered before and therefore ChatGPT was asked to give a brief explanation of these 2 metrics.



what does "mAP@0.5: 0.7148494034225592" mean

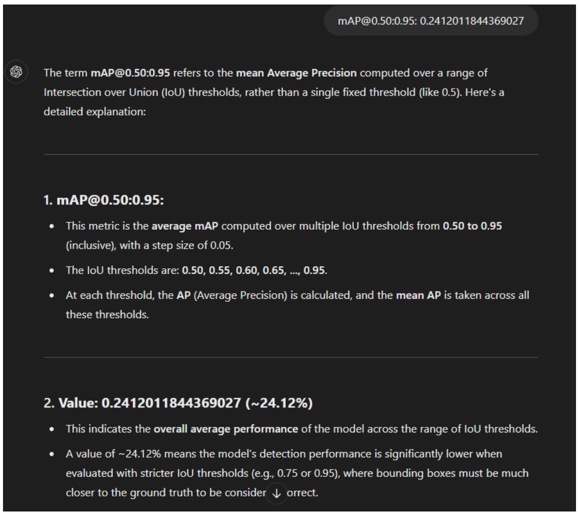
The term **"mAP@0.5: 0.7148494034225592"** is commonly used in the context of evaluating object detection models, such as those trained with frameworks like YOLO, Faster R-CNN, or SSD. Here's a breakdown of its meaning:

1. mAP (Mean Average Precision):

- **AP (Average Precision):** A metric that evaluates the precision-recall curve for a single class. It is the area under the precision-recall curve (AUC-PR).
- **mAP (Mean Average Precision):** The mean of the AP scores across all classes in the dataset. It provides a single scalar value summarizing the performance of the model for all classes.

2. @0.5:

- The `@0.5` indicates the **Intersection over Union (IoU) threshold** used during evaluation.
- IoU measures the overlap between the predicted bounding box and the ground truth bounding box.
- At an IoU threshold of 0.5, a prediction is considered correct if the IoU between the predicted and ground truth bounding boxes is **at le ↓ 0.5** (50% overlap).



mAP@0.50:0.95: 0.2412011844369027

The term **mAP@0.50:0.95** refers to the **mean Average Precision** computed over a range of Intersection over Union (IoU) thresholds, rather than a single fixed threshold (like 0.5). Here's a detailed explanation:

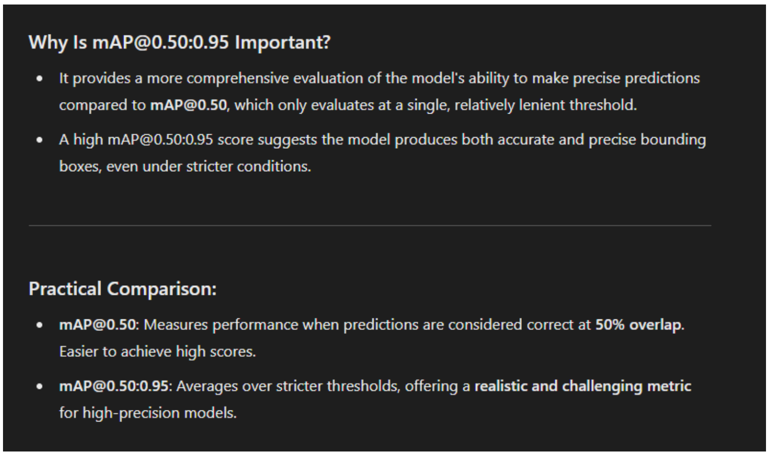
1. mAP@0.50:0.95:

- This metric is the **average mAP** computed over multiple IoU thresholds from **0.50 to 0.95** (inclusive), with a step size of 0.05.
- The IoU thresholds are: **0.50, 0.55, 0.60, 0.65, ..., 0.95**.
- At each threshold, the **AP** (Average Precision) is calculated, and the **mean AP** is taken across all these thresholds.

2. Value: 0.2412011844369027 (~24.12%)

- This indicates the **overall average performance** of the model across the range of IoU thresholds.
- A value of ~24.12% means the model's detection performance is significantly lower when evaluated with stricter IoU thresholds (e.g., 0.75 or 0.95), where bounding boxes must be much closer to the ground truth to be consider ↓ orrect.

Figure 3: ChatGPT Example Prompt 3   Figure 4: ChatGPT Example Prompt 4



Why Is mAP@0.50:0.95 Important?

- It provides a more comprehensive evaluation of the model's ability to make precise predictions compared to **mAP@0.50**, which only evaluates at a single, relatively lenient threshold.
- A high mAP@0.50:0.95 score suggests the model produces both accurate and precise bounding boxes, even under stricter conditions.

Practical Comparison:

- **mAP@0.50**: Measures performance when predictions are considered correct at **50% overlap**. Easier to achieve high scores.
- **mAP@0.50:0.95**: Averages over stricter thresholds, offering a **realistic and challenging metric** for high-precision models.

Figure 5: ChatGPT Example Prompt 5

Improvements, Errors and Contributions

The use of generative AI in our project played a key role in addressing the challenges that were encountered both throughout model development and training, and documentation writing. ChatGPT was instrumental in providing suggestions aimed at improving code efficiency through the responses it generated. It also helped when it came to troubleshooting any bugs that occurred during development, such as incompatibility with specific package versions or parameters not being passed correctly.

During the writing of the documentation report, ChatGPT helped tremendously when it came to improving the readability and clarity of the report while maintaining the necessary structure. It also assisted when it came to evaluating the results produced by the models, by providing insights into which model performed the best depending on the given metrics and why.

Individual Reflection

Matthias's Reflection:

The use of Generative AI throughout this project, more notably ChatGPT, has been extremely beneficial, as it allowed me to produce more efficient and polished work, while receiving immediate feedback regarding any possible improvements that could be made, especially when training the object detector. Additionally, this tool allowed me to improve my writing skills, especially when it comes to producing clear text with proper grammar and structure. This enabled the documentation to be more cohesive with adequate explanations. Although the use of ChatGPT proved to be quite useful, as it helped me improve my performance, special care was given to research a bit to ensure that the suggestions provided by the AI were suitable for the given task before eventually applying them.

Jacob’s Reflection:

I used ChatGPT to debug my code. When I got an error where the Google Colabaratory stopped being able to navigate files in my google drive, by simply copying and pasting the error message into ChatGPT, it outputted the solution. This was way faster than if I tried searching the internet for anyone else who had the same problem. I learned that when using ChatGPT specifically in the context of solving errors, there is no need to waste time and provide context for the error, usually just copy pasting the error message is enough since it is a LLM and can't understand anything anyway. ChatGPT also generated the line of code which starts the training of the model, as I have found that ChatGPT is an effective tool for fixing the structure of one line worth of code.

Kyle’s Reflection:

ChatGPT was used from my end to clear and clarify any doubts I had while trying to understand which Convolutional Neural Network to pick. It provided very straight-to-the-point answers and when I was still unsure, I asked it to re-explain that part and it performed as expected. Double-checking with non-AI sources such as YOLO official documentation was done to ensure that ChatGPT was not hallucinating especially when asking questions about parameters such as output image size and certain metrics like mAP (Mean Average Precision). Whilst AI is a good tool for one to use, one must remember not to depend 100% on it and always verify your responses with secondary or more credible official sources. Google Colab also has Gemini Assistance built into their user interface which was also fairly useful in suggesting code fixes to Jupyter Notebook code, since it has access to all the Notebook’s code at once unlike ChatGPT which only has access to the prompt you ask to it and the user may or may not explain him/herself in the right way.

References and List of Resources Used

[1] “Introducing GPT-4o and more tools to ChatGPT free users,” Openai.com, 2024. https://openai.com/index/gpt-4o-and-more-tools-to-chatgpt-free/