
Advanced Computer Vision for AI Project - GenAI Journal

Authors: Luca Bugeja, Matthias Ellul, Remi Heijmans, Liam Debono

Date: January 2026

1. Introduction

Throughout the process of completing this project, the GenAI tool used included ChatGPT. This tool was chosen specifically due to using it for coding purposes. From personal experience, the newer version of ChatGPT performs well in coding scenarios, while having no limits on its use. While Claude is often hailed as the best coding tool, it has two main drawbacks. Firstly, the limit for the use of the free version is too little, and additionally, unless prompted specifically multiple times, I find that the code is over-complicated for what needs to be accomplished. Therefore, I settled for ChatGPT. Luckily, a free version of the plus allowing use of ChatGPT 5.2 was rolled out and I took advantage of this. (Luca Bugeja)

Generative Artificial Intelligence (GenAI) tools were used throughout this project as supportive aids rather than as replacements for technical understanding or implementation. When seeking information, several tools were used in the aim of mitigating bias by considering the output of more than one model, including OpenAI's ChatGPT, Anthropic's Claude Opus 4.5, and Google's Gemini. However, the primary tool utilised was ChatGPT, chosen for its ability to assist with technical clarification, debugging, documentation drafting, and critical reflection. The motivation for using generative AI was to improve efficiency, clarity, and correctness during the development and documentation phases of the project. GenAI was not used to automatically generate datasets, annotations, or final model outputs. All model training, evaluation, and experimental results were produced manually by the author. (Matthias Ellul)

For my individual tasks, Google Gemini 3 was used as a technical assistant. The AI assisted in the initial setup of tasks to give me an easy start. Later utilized for data cleaning, code refactoring , and ensuring the final report maintained a professional academic tone through automated spell-checking and logic verification.(Remi Heijmans)

2. Ethical Considerations

In terms of ethical considerations, there are various which need to be made in the case that images were given to the chatbot, which in my case was not done. Since we do not know where images given to GenAI tools end up with certainty, this has to be done with careful consideration. If it were to be carried out, the images should be the blurred versions so that they comply with GDPR. In my case, the use of the tool was purely for coding purposes, and as for prompts, the only information provided was text based of my code. A possibility of biased code may be present, including for instance, biased metrics. However, in this case, the metrics and models used were standard metrics used in computer vision projects such as mAP and Precision. (Luca Bugeja)

The use of generative AI in this project was guided by ethical principles related to academic integrity, data privacy, and transparency. Firstly, generative AI was not used to fabricate experimental results, inflate performance metrics, or bypass learning outcomes. All quantitative results reported in the technical documentation originate from independently executed experiments. GenAI outputs were treated as suggestions that required verification and, where necessary, correction. Secondly, privacy and data protection considerations were upheld. No raw images containing identifiable individuals, licence plates, or sensitive information were shared with generative AI systems. Dataset handling remained compliant with GDPR requirements, and GenAI tools were only used on abstracted descriptions of the data and results. Finally, transparency was maintained by explicitly documenting the use of generative AI in this appendix. This ensures accountability and allows evaluators to distinguish between human-authored work and AI-assisted refinement. (Matthias Ellul)

In accordance with GDPR and data privacy standards, I ensured that no raw or identifying images from the traffic sign dataset were uploaded to the GenAI interface. My interaction with the tool was strictly text-based, providing abstracted descriptions of code structures and error logs rather than visual data. This approach mitigated the risk of unauthorized data retention by the AI provider while allowing for effective technical troubleshooting. Furthermore, the performance metrics and model outputs reported were generated through independent execution in a secure Google Colab environment, ensuring that the results remain transparent, reproducible, and ethically sound.(Remi Heijmans)

3. Methodology

Once the dataset was wholly gathered, the python notebooks which detect the sign type and the attribute were to be made. For coding therefore, the workflow included coding in one window on Jupyter notebooks, while on another, ChatGPT was held open to help. While other tools like VS Code allow for the use of GenAI tools within the application, I find that I am more comfortable in keeping one tab for the tool and one for the notebook.

Generative AI was primarily integrated into the workflow during the following stages:

1. Technical clarification

GenAI was used to clarify concepts related to object detection metrics (e.g., mAP@50 vs. mAP@50–95), CNN architectures, and dataset imbalance interpretation.

2. Debugging and implementation support

When encountering errors in Jupyter notebooks (e.g., model saving issues or metric computation bugs), GenAI was used to propose potential fixes, which were then manually tested, modified according to the needs of the project, and validated.

3. Documentation drafting and refinement

Draft text for documentation sections was generated and iteratively refined to ensure academic tone, logical structure, and alignment with marking criteria.

4. Critical reflection and interpretation

GenAI assisted in articulating limitations, error analysis, and future work in a structured and academically appropriate manner.

At all stages, outputs were reviewed critically and adapted to reflect the actual implementation and results of the project.

4. Prompts and Responses

(Luca Bugeja)

An issue I had was handling the libraries and using the GPU successfully in training the models. I found ChatGPT was very useful in this regard. Since I was using Python 3.13, and was faced with several issues with the necessary libraries, such as torch cu128 for the GPU, I had prompted this issue and it was suggested that I create a separate environment for this project. I followed through, using Python v3.11 and installed all the necessary libraries and the issue was solved. As for general coding, one instance in which it helped with was the plotting portion

2 Display YOLO's auto-generated plots (quickest)

In a notebook cell:

```
python Copy code
from PIL import Image
import matplotlib.pyplot as plt
from pathlib import Path

RUN_DIR = Path("runs_sign_type/yolov11_sign_type4")

for name in [
    "results.png",
    "confusion_matrix.png",
    "PR_curve.png",
    "P_curve.png",
    "R_curve.png",
    "F1_curve.png",
]:
    img = Image.open(RUN_DIR / name)
    plt.figure(figsize=(6,6))
    plt.imshow(img)
    plt.axis("off")
    plt.title(name)
    plt.show()
```



Okay now everything worked, how do I plot the results now

Great 👍 — once training finishes successfully, Ultralytics already **saves everything for you**. You can both **view the saved plots and plot metrics yourself** in the notebook.

Below is the clean, standard way to do both.

3 Where YOLO saved your results

From your logs:

```
bash Copy code
runs_sign_type/yolov11_sign_type4/
```

Inside that folder you'll find:

- `results.csv` ← metrics per epoch
- `results.png` ← auto summary plot
- `confusion_matrix.png`
- `PR_curve.png`, `P_curve.png`, `R_curve.png`, `F1_curve.png`
- `labels.jpg`
- `weights/best.pt`, `weights/last.pt`



As visualized above, the location of the created plots was given and the code which plotted these graphs was also provided. In this case, the names of the actual plots were not completely correct and had to be amended, however with this fix, everything worked.

Another instance in coding in which I used the chat-bot was to obtain the hyper-parameters for training the YOLO11 model. In this case, standard hyperparameters were given, which produced a good result.

Additionally, another prompt which I used was to obtain the precision, recall and F1-score of the trained model on the test set, both as a whole and per-class basis. This was done to evaluate better the model's performance. In this case, multiple prompts were given to refine the answer. The first outcome included IoU matching, which I was not looking for. After re-prompting, a generic count of the predictions and comparison with the total ground truth numbers was given, which also was not satisfactory. Finally, after refining the prompt, a correct answer was given which needed a few changes. The metrics provided also included micro precision and macro precision (weighted) which proved to be useful.

(Matthias Ellul)

The following examples illustrate some notable prompts used by Matthias during the project and how their responses contributed to improving the work.

Prompt 1: Technical Interpretation

Prompt:

"Explain how to interpret mAP@50 and mAP@50–95 for a YOLOv8 traffic sign detector."

Contribution (Summary):

Provided a detailed but straightforward explanation that helped clarify how stricter IoU thresholds affect mAP scores, enabling a more accurate and confident explanation of detection performance in the results section.

Prompt 2: Documentation Structuring

Prompt:

"Provide some references that can help me write an academic background section explaining YOLOv8 and MobileNetV2 for a computer vision project."

Contribution (Summary):

Apart from providing the requested references, ChatGPT even began drafting a technically accurate background section itself, including relevant points that it extracted from the original sources. This was then expanded with citations and refined to match IEEE academic standards.

Prompt 3: Error Analysis Support

Prompt:

“My classifier performs poorly on the ‘Damaged’ class. How should this be explained academically?”

Contribution (Summary):

The response attributed dataset imbalance and class ambiguity as the root causes. It provided suggestions on how such limitations can be minimised in any future work.

Prompt 4: Debugging a Model Saving Error (TensorFlow)

Prompt:

“My TensorFlow Keras model fails when calling `model.save()` in a Jupyter notebook. What are common causes and how can this be fixed?”

Contribution (Summary):

The response suggested checking the save path format, verifying TensorFlow version compatibility, and ensuring that the correct save method was used. It also highlighted potential issues with chained method calls and filesystem permissions. This prompt directly supported resolving a runtime error encountered during model persistence. The guidance helped identify improper usage of the save function and ensured the trained classifier could be exported correctly for later evaluation.

Prompt 5: Understanding Detection Metrics in Code

Prompt:

“In YOLOv8 validation output, how are true positives, false positives, and false negatives calculated, and how do these relate to IoU thresholds?”

Contribution (Summary):

The response explained IoU-based matching between predicted and ground-truth boxes and clarified how confidence thresholds affect TP, FP, and FN counts. This explanation enabled correct interpretation of validation statistics and supported accurate reporting of detection counts in the results section, avoiding common misinterpretations of object detection metrics.

(Remi Heijmans)

Prompt 6: Implementing a Two-Stage Detector (Task 2A)

Prompt: "I need to implement a Faster R-CNN model in PyTorch for Task 2A to compare against a YOLO model. How do I replace the pre-trained head of a ResNet-50 Faster R-CNN to detect 6 specific sign classes plus a background class?"

Contribution: The response provided the logic to swap the standard FastRCNNPredictor with a custom one tailored to the project's 7-class requirement. This support ensured that the "Alternative Architecture" requirement of the project was met with a technically sound Two-Stage model implementation.

Prompt 7: Final Document Refinement

Prompt: "I have drafted my individual sections for the project report. Can you perform a spell-check and rewrite these paragraphs to ensure they use a formal academic tone suitable for a university-level Computer Vision assignment?"

Contribution: The AI assisted in refining the language of the "Methodology" and "Individual Reflection" sections, ensuring clarity, professional terminology, and alignment with the marking criteria.

5. Improvements, Errors and Contributions

(Luca Bugeja)

In this project, I believe Generative AI helped a lot in the coding phase. As for the dataset formulation part, which constituted the largest amount of work, I did not use the tool, since I did not see any value in doing so. As for the coding phase, including visualization, training, evaluation and comparison of the detection models, it proved useful. Tedious issues such as the library problem I had mentioned prior, are much more easily resolved using generative AI, since actionable outputs are given and are in general quite reliable from experience. As for general coding, I also found the tool helpful. While in certain instances, such as when prompted for training hyper-parameters, a good answer was given immediately, in other cases (such as the evaluation), repetitive prompting was necessary to obtain a satisfactory answer. In general, the code given still had to be edited in parts but the provided code was still very useful.

(Matthias Ellul)

Generative AI contributed positively by:

- Improving the clarity and structure of documentation
- Reducing time spent on syntax-level debugging
- Supporting accurate metric interpretation
- Helping maintain a professional academic tone

However, limitations were also observed. Some responses:

- Overgeneralised performance claims
- Assumed larger datasets than those actually used
- Suggested techniques outside the project scope

- Provided code that contained logical and sometimes even syntax errors

These issues required manual correction, reinforcing the importance of critical evaluation and never relying blindly on generative AI outputs.

6. Individual Reflection

(Luca Bugeja)

On the whole, I think that while GenAI was only used for a portion of the project's work due to obvious limitations in dataset gathering and formulation for instance, it still proved to be a valuable tool in other cases. Personally, while I believe that it increases efficiency drastically in coding, since the output provided generally resembles or is what is needed closely, minor edits at the least still have to be frequently made even with repetitive prompting. In addition, this saves time since starting from scratch would take much longer. As mentioned prior, I believe that it was most helpful in my case for library integration, and general coding purposes.

(Matthias Ellul)

The use of generative AI significantly improved efficiency during this project, particularly in documentation and debugging. Tasks that would normally require extensive searching across documentation or forums were completed more quickly, allowing more time to focus on experimentation and analysis. One key insight gained was that generative AI is most effective when used as a collaborative assistant rather than an authoritative source. The quality of output was strongly dependent on prompt specificity and the user's own understanding of the subject matter.

Overall, generative AI proved most helpful for:

- Documentation drafting
- Metric interpretation
- Code debugging explanations

It was less effective for:

- Dataset-specific decisions
- Architectural design choices
- Performance optimisation

This experience reinforced the importance of domain knowledge and critical thinking when integrating generative AI into academic workflows.

7. References and Resources Used

- OpenAI, *ChatGPT GPT-5.2*, <https://chat.openai.com>
- Anthropic, *Claude Sonnet 4.5*, <https://claude.ai>
- Google, *Gemini 3*, gemini.google.com