Brief Report

Introduction
The primary objective of this project was to detect changes in scenarios captured in ***`test.npz`*** compared to ***`base.npz`***. To achieve this, image processing techniques and deep learning were employed, training a deep neural network to detect objects in images from these distinct datasets. After detection, the changes between the scenarios represented by the images were identified and analysed, allowing for precise and efficient comparison of the detections.

Neural Network Training
To perform object detection and, consequently, identify changes between the scenarios in ***`test.npz`*** and ***`base.npz`***, it was necessary to train a deep neural network, YOLOv8. The choice of YOLOv8 was motivated by its high accuracy and efficiency in real-time object detection, which is crucial for the rapid and precise identification of changes between scenarios. The use of Ubuntu 20.04 with CUDA was decided to leverage the hardware acceleration provided by GPUs, significantly reducing the model training time. The ***labelImg*** tool was chosen for image labelling due to its simplicity and effectiveness in preparing data for training object detection models.

Within the project, there is a directory named ***`train`*** that contains images in *.png* format extracted from the provided *.npz* files. These images were then labelled using the ***labelImg*** tool for neural network training. The steps for training included:

1. Data Preparation	2. Environment Setup	3. Network Training	4. Model Evaluation
• Extraction of images in *.png* format from the *.npz* files. • Labelling with ***labelImg***.	• Ubuntu 20.04 with CUDA to utilise GPU during training, accelerating the process.	• Training YOLOv8 using the labelled data, adjusting hyperparameters.	• Post-training, the model was validated to check accuracy.

Within the ***`train`*** directory, there is also a subdirectory named ***`results_custom`***, which contains all the results generated by the YOLOv8 framework, including the trained model weights, confusion matrices, performance metrics, and other important artefacts for model evaluation and validation.

System Development
For the development of the system, three Python scripts were created: one basic script, one using *Streamlit*, and another using *Flask*. The basic script allows the visualisation of detections directly in the operating system, while the *Streamlit* script provides a quick and easy-to-use web interface for analysing results. The *Flask* script offers a more robust and interactive web interface, enabling remote access and convenient sharing of results. Detailed instructions for executing these scripts are available in the project's ***`README.md`*** file.

Proposed Improvements with More Time
If we had an additional six months to work on the project, the following improvements could be implemented: enhance the model's accuracy by refining training hyperparameters and increasing dataset diversity; implement data augmentation techniques to create variations in the training images, helping the network generalise better; develop a more sophisticated web interface to offer a richer user experience, including interactive visualisations and detailed analysis of detections; explore integration with other data analysis tools and deep learning frameworks to expand the system's functionalities; and automate the labelling process by developing scripts to reduce the time and effort required to prepare data.

Important Note
To set up the development environment and install the necessary dependencies, please refer to the ***`README.md`*** file in the project's root directory.