National College of Ireland

Configuration Manual

MSc Research Project
Artificial Intelligence

Adam Lysaght
Student ID: x23314869

School of Computing
National College of Ireland

Supervisor: Devanshu Anand

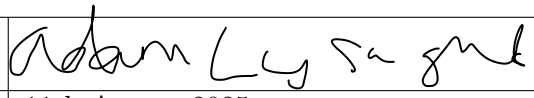
National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Adam Lysaght
Student ID:	x23314869
Programme:	Artificial Intelligence
Year:	2025
Module:	MSc Research Project
Supervisor:	Devanshu Anand
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	XXX
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:	11th August 2025

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

Adam Lysaght
x23314869

1 Introduction

This configuration manual provides guidance on how to replicate the multi-modal accident hotspot prediction framework. You can run this framework locally or on cloud platforms like Google Cloud Platform (GCP).

2 System Requirements

2.1 Hardware Requirements

- CPU: 8+ cores
- RAM: 32GB
- GPU: NVIDIA GPU with 16GB+ VRAM
- Storage: 100GB+ SSD

2.2 Software Requirements

The project requires Python 3.8+ with machine learning libraries. GPU support is essential for training the deep learning models:

- Python 3.8+ (3.9 recommended)
- NVIDIA CUDA 11.8+ and cuDNN 8.6+ for GPU acceleration
- All Python dependencies are listed in `requirements.txt` (installed automatically in next section)

3 Setup

3.1 Clone the Repository

First, clone the project from GitHub:

```
git clone https://github.com/adam1100/mscai.git
cd mscai
```

3.2 Setup Python Environment

Create a virtual environment to isolate the project dependencies:

```
python -m venv tf-env
tf-env\Scripts\activate
```

Install all required Python packages: This installs TensorFlow, Keras, scikit-learn, and other dependencies needed for computer vision and machine learning tasks.

```
pip install -r requirements.txt
```

Add your Python path of the directory to the env file

```
/env
```

3.3 Directory Structure

The project contains following directory structure:

```
mscai/
|--- data/
|    |--- raw/              # Raw collision CSV (included in repo)
|    |--- processed/        # Cleaned datasets (generated)
|    |--- sv/               # Street view images (downloaded via API)
|    |--- sat/              # Satellite imagery (downloaded via API)
|--- src/                   # Source code / execution files
|--- test_output/           # Experiment outputs
|--- results/               # Model training results
```

4 Data Setup

4.1 Data

The street view and satellite images are **not included in the GitHub repository** due to size constraints. These can be downloaded using the provided data sourcing scripts. The collision data CSV file is included in the repository.

4.2 API Configuration

To download the street view and satellite images, you need to configure API access. Create API keys as follows:

- Go to Google Cloud Console
- Enable the Street View Static API & Maps Static API
- Create an API key and add it to `src/config/gcp_config.py` and `.env`
- The system uses this to download street view images from specific coordinates

5 Running the Framework

The framework follows a multi-step pipeline from raw collision data to trained models.

5.1 Step 1: Prepare the Collision Data

Start by processing the raw UK collision data to identify accident hotspots:

```
experiments/clustering.ipynb
python -m src.data_processing.data_preprocessing
```

This script loads the collision CSV file, cleans the data, and uses DBSCAN clustering to identify accident hotspots. It creates binary labels (hotspot vs non-hotspot) that the models will learn to predict. The output is stored in `data/processed/`.

5.2 Step 2: Download Street View and Satellite Images

Since images aren't in the repository, download them from the APIs:

```
experiments/image_sourcing.ipynb
python -m src.data_processing.street_view_sourcing
python -m src.data_processing.satellite_sourcing
```

The street view sourcing script takes each accident location and downloads 4 directional images (north, east, south, west views). The satellite sourcing script downloads overhead imagery for each accident locations. This can take several hours depending on the number of locations. Images are saved locally so you only need to run this once.

5.3 Step 3: Extract Features

Transform the raw images into machine learning-ready features:

```
experiments/feature_extraction copy.ipynb
python -m src.utils.enhanced_pipeline_validation (Streetview)
python -m src.utils.simple_satellite_pipeline (Satellite)
```

This script processes all downloaded images through pre-trained EfficientNetB0/ResNet50 neural networks to extract visual features. It handles the conversion from high-resolution images (640×640) to feature tensors that capture important visual patterns. The features are saved as numpy arrays for fast loading during model training. Note these arrays can take up alot of memory (5GB+) depending on the data size.

5.4 Step 4: Train the Models

Train different types of models on your extracted features data. The framework supports three approaches:

Street View - Machine Learning & Deep Learning Models:

```
experiments/unified_modeling_notebook.ipynb
python -m src.models.sequence_modelling3 --data_type=streetview
```

Satellite - Machine Learning & Deep Learning Models:

```
experiments/unified_modeling_notebook.ipynb
python -m src.models.sequence_modelling3 --data_type=satellite
```

Combined - Deep Learning Models:

```
experiments/unified_modeling_notebook.ipynb
experiments/create_multimodal_dataset.ipynb
python -m src.models.sequence_modelling3 --mode=combined
```

5.5 Step 5: View Results

After training completes, results are saved in the `results/` directory. Each run generates:

- JSON files with detailed performance metrics
- Model comparison tables
- Training history plots
- Confusion matrices showing prediction accuracy

```
experiments/unified_modeling_notebook.ipynb
```

Jupyter: Run the Jupyter notebooks in `experiments/` to visualize and process through the steps interactively.

6 Common Issues

Here are solutions to common problems you might encounter:

Out of Memory Errors: GPU memory errors during model training - Reduce the batch size in the training scripts or use a GPU with more memory.

Missing Dependencies: Activate virtual environment and install all requirements: `source tf-env/bin/activate` then `pip install -r requirements.txt`.