National College of Ireland

Configuration Manual

MSc Research Practicum
Data Analytics

Balram Bhukya
Student ID: x23245069

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	……. ……Balram Bhukya……
Student ID:	……x23245069…….……
Programme:	……Data Analytics…… **Year:** 2024-2025
Module:	……Research Practicum…….……
Lecturer:	……Hicham Rifai…….……
Submission Due Date:	……September 15th, 2025……
Project Title:	……Multi-Sensor Fusion-Based Anomaly Detection in Simulated CAV Environments Using a D-CNN-LSTM Autoencoder…….……
Word Count:	7642 **Page Count :** 20

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: ……Balram Bhukya……

Date: ……15/09/2025……

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

Balram Bhukya
Student ID: x23245069

1 Introduction

This document outlines a thorough technical reference for setting up an appropriate environment and running the computational experiments described in the research project "Multi-Sensor Fusion-Based Anomaly Detection in Simulated CAV Environments Using a D-CNN-LSTM Autoencoder". This document is meant to provide a level of detail so that the research can be recapitulated in full, specifically indicating the appropriate hardware, software, data organization, and procedure/steps to execute as also provide key snippets of code for reference.

The project falls under two main stages, which exist in two separate Jupyter Notebooks. The first stage pertains to preprocessing the raw D2CAV dataset into a balanced dataset. The second stage is a large workflow of feature engineering, dimensionality reduction, training nine different models, and evaluating the models. A user will be able to follow these instructions to illustrate the entire experimental pipeline and produce all final model artifacts.

2 System Requirements

2.1 Hardware

- **CPU:** Multi-core 64-bit processor (e.g., Intel Core i7 or AMD Ryzen 7).
- **RAM:** Minimum of 16 GB.
- **GPU:** NVIDIA GPU with CUDA support (6 GB+ VRAM) is strongly recommended to accelerate deep learning model training.
- **Storage:** Minimum of 15 GB of free disk space.

2.2 Software

- **OS:** Windows 10/11, macOS 11+, or Ubuntu 20.04+.
- **Python:** Version 3.8, 3.9, or 3.10.
- **Libraries:** tensorflow, pandas, numpy, scikit-learn, matplotlib, seaborn, plotly, joblib.

3 Environment Setup

3.1 Data Acquisition and Placement

This manual assumes you have the project files. It is critical to place the dataset directory and the two Jupyter notebooks (balanced-dataset-generation.ipynb, modelling.ipynb) in a single root folder. The expected structure is:

```
project_root/
├── dataset/
│   ├── hard_brake/
│   ├── lane_left/
│   └── ... (other maneuver folders)
├── balanced-dataset-generation.ipynb
└── modelling.ipynb
```

3.2 Create and Activate Virtual Environment

Execute the following commands in your terminal from the project_root directory to create and activate an isolated Python environment.

```
# Create the virtual environment
python -m venv

# Activate the environment
# On Windows: .\venv\Scripts\activate
# On macOS/Linux: source venv/bin/activate
```

3.3 Install Dependencies

Create a file named requirements.txt in the project_root directory with the following content:

```
tensorflow==2.10.0
pandas==1.5.3
numpy==1.23.5
scikit-learn==1.3.0
matplotlib==3.6.3
seaborn==0.12.2
plotly==5.11.0
joblib==1.2.0
notebook
```

Install all packages with a single command:

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

4 Execution Workflow

The project must be executed in two sequential stages.

4.1 Stage 1: Dataset Generation and Balancing

This stage processes the raw CSV files into a single, balanced dataset ready for machine learning.

1. **Launch Jupyter:** From your activated terminal, start the Jupyter server:

```
jupyter notebook
```

2. **Run Preprocessing Notebook:** Open balanced-dataset-generation.ipynb and execute all cells sequentially (**Cell > Run All**). The notebook automates the process of reading, labeling, and downsampling the data.
3. **Code Highlight - Data Aggregation:** The core of this script reads all CSV files from the subdirectories, assigns a class label, and concatenates them into a single DataFrame.

```
import pandas as pd
import os
import glob
from sklearn.utils import resample

# Find all CSV files recursively within the dataset directory
all_files = glob.glob(os.path.join('dataset', '**', '*.csv'), recursive=True)

df_list = []
for file in all_files:
    # Extract the class name from the parent directory's name
    class_name = os.path.basename(os.path.dirname(file))
    temp_df = pd.read_csv(file)
```

```
    temp_df['class'] = class_name
    df_list.append(temp_df)

# Combine all individual dataframes into one
full_df = pd.concat(df_list, ignore_index=True)

# The rest of the notebook performs balancing and saves the output.
```

4. **Expected Outcome:** This script will create a new directory named processed_data/ containing the clean input files for the next stage: balanced_dataset.csv, features.csv, and labels.csv.

4.2 Stage 2: Model Training and Evaluation

This stage executes the main experimental pipeline, from feature engineering to model saving.

1. **Prerequisite:** Verify that the processed_data/ directory exists from Stage 1.
2. **Run Modeling Notebook:** Open modelling.ipynb in Jupyter and execute all cells (**Cell > Run All**). This is a long process, especially the model training cells.
3. **Code Highlight - PCA Dimensionality Reduction:** After extensive feature engineering, PCA is used to reduce the data from 36 dimensions to 8, while retaining 95% of the variance.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Standardize features before applying PCA
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features_engineered)

# Initialize PCA to retain 95% of the variance
pca_optimal = PCA(n_components=0.95, random_state=42)

# Fit and transform the data
features_pca = pca_optimal.fit_transform(features_scaled)
print(f"Original shape: {features_engineered.shape}")
print(f"PCA shape: {features_pca.shape}")
```

4. **Code Highlight - D-CNN-LSTM Autoencoder Training:** The core model is trained in an unsupervised manner, using only "normal" data as both input and target. This teaches the model to reconstruct normal driving patterns.

```
# Prepare data for autoencoder (reshape for CNN)
X_train_reshaped = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)

# Train only on normal data for anomaly detection
normal_indices = y_train == 0
X_train_normal = X_train_reshaped[normal_indices]

# Train the autoencoder
history = autoencoder.fit(
    X_train_normal, X_train_normal,  # Input and target are the same
    epochs=50,
    batch_size=32,
    validation_data=(X_val_reshaped[y_val == 0], X_val_reshaped[y_val == 0]),
    callbacks=[early_stopping, reduce_lr]
)
```

5. **Expected Outcome:** A new directory named saved_models/ will be created, containing all final artifacts: .pkl files for scikit-learn models and scalers, .h5 files for TensorFlow models, and a .csv file with the final performance metrics.

5 Verification of Saved Artifacts

The final cell in modelling.ipynb already performs a verification test. This test ensures the saved models and preprocessing components can be loaded and used for inference on new data. The logic involves loading the scaler, pca_model, and a trained model, then passing a sample of mock data through the entire pipeline.

Verification Code Snippet:

```
import joblib
import numpy as np
import pandas as pd
from tensorflow.keras.models import load_model

def test_saved_models(save_dir, timestamp, X_test_sample):
    """Test if saved models can be loaded and make predictions"""
    print("=== TESTING SAVED MODELS ===")
    try:
        # Load preprocessing components
        loaded_scaler = joblib.load(f'{save_dir}/scaler_{timestamp}.pkl')
        loaded_pca = joblib.load(f'{save_dir}/pca_model_{timestamp}.pkl')

        # Test preprocessing
        X_scaled = loaded_scaler.transform(X_test_sample)
        X_pca = loaded_pca.transform(X_scaled)
        print("✓ Preprocessing components working")

        # Load and test autoencoder
        loaded_autoencoder =
load_model(f'{save_dir}/dcnn_lstm_autoencoder_{timestamp}.h5', compile=False)
        X_reshaped = X_pca.reshape(X_pca.shape[0], X_pca.shape[1], 1)
        reconstruction = loaded_autoencoder.predict(X_reshaped, verbose=0)
        print("✓ Autoencoder prediction successful")

    except Exception as e:
        print(f"✗ Error testing models: {str(e)}")

# This test is run at the end of the notebook.
# test_saved_models(save_dir, timestamp, features_engineered.iloc[:5])
```

6 Conclusion

By following this guide, you have now set up the project environment, processed raw data, and completed the entire model training and evaluation pipeline. The reproducible artifacts are located in the saved_models/ directory. The verification script successfully ran and verified that your artifacts are functional and can be used for inference or further analysis.