Configuration Manual

MSc Research Project

Artificial Intelligence

Santhosh Rajan

Student ID: 23318881

School of Computing

National College of Ireland

Supervisor: Lavish Thomas

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Santhosh Rajan
Student ID:	23318881
Programme:	Artificial Intelligence
Year:	Sept 2024 - 2025
Module:	MSc Research Project
Supervisor:	Lavish Thomas
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	822
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:	P. Santhosh
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

Santhosh Rajan

23318881

1. Introduction and Scope

This configuration manual serves to provide a thorough record of the computing environment, software dependencies, and organisational context required to replicate the experiments and results of the fall-detection system implemented for this research.

The intent is to allow another researcher or practitioner to duplicate the system with as little ambiguity as possible for the purposes of academic validation, further development, or deployment in practice. While the document focuses on the current implementation, it does also consider best practices for extending the system into a more modular, production-ready solution.

2. System Environment

2.1 Operating System and Kernel

- **Distribution:** Ubuntu 24.04.2 LTS (“noble”)
- **Kernel:** Linux 6.14.0-27-generic, PREEMPT_DYNAMIC, x86_64 architecture
- **Session Type:** Wayland display protocol ($XDG_SESSION_TYPE=wayland)
- **Locale:** en_US.UTF-8
- **Time Zone:** Europe/Dublin (Irish Standard Time, +01:00)

2.2 Hardware Summary

- **Processor:** AMD Ryzen 7 PRO 7840U with integrated Radeon 780M graphics
  - 8 physical cores / 16 threads
  - Maximum clock frequency: ~5.13 GHz
  - AVX2 and AVX-512 instruction sets supported (including BF16 and VNNI)
- **Memory:** 30 GiB RAM (available), 8 GiB swap space
- **Graphics Adapter:** AMD Phoenix1 (Radeon 780M)
- **Storage:** Sufficient storage space for datasets and intermediate results (minimum 10 GB free recommended for datasets and models)

3. Python Environment

3.1 Interpreter and Package Manager

- **Python Version:** 3.12.7

- **Pip Version:** 25.1.1
- **Environment Management:** Conda and virtualenv both available, experiments were run in a Conda-managed environment.

3.2 Required Packages

Only the packages required for executing this project are mentioned here. This helps avoid unnecessary dependencies and ensures reproducibility.

Core Scientific Libraries:

- numpy 1.26.4
- scipy 1.13.1
- pandas 2.2.2

Machine Learning Frameworks:

- scikit-learn 1.5.1
- tensorflow 2.19.0
- tf_keras 2.19.0
- torch 2.7.0 (optional used for comparative tests)
- xgboost 3.0.0 (optional used for baseline gradient boosting experiments)

Visualisation Tools:

- matplotlib 3.9.2
- seaborn 0.13.2

Utility:

- tqdm 4.66.5

Jupyter Environment:

- jupyter 1.0.0
- ipykernel 6.28.0
- notebook 7.2.2

Optional NLP Packages (if voice-based interaction is enabled):

- transformers 4.51.3
- tokenizers 0.21.1
- sentencepiece 0.2.0
- huggingface-hub 0.30.2

4. Environment Setup

A clean environment is strongly recommended for reproducing this research project work.

Using virtualenv:

```
python3 -m venv .venv
source .venv/bin/activate
pip install --upgrade pip==25.1.1
pip install numpy==1.26.4 scipy==1.13.1 pandas==2.2.2 \
scikit-learn==1.5.1 tensorflow==2.19.0 tf_keras==2.19.0 \
matplotlib==3.9.2 seaborn==0.13.2 tqdm==4.66.5 \
jupyter==1.0.0 ipykernel==6.28.0 notebook==7.2.2
```

Using Conda:

```
conda create -n fallguard python=3.12 -y
conda activate fallguard
pip install --upgrade pip==25.1.1
# Install packages as above
```

Once installed, the environment configuration should be frozen with:

```
pip freeze > requirements_exact.txt
```

5. Project Layout

5.1 Recommended Modular Layout

For a production-scale system with additional features such as **voice recognition**, **alert systems**, and **integration with smart home devices**, a more modular layout is advisable:

```
project-root/
├── data/
│   ├── raw/
│   └── processed/
├── src/
│   ├── voice_recognition/
│   ├── alert_system/
│   ├── data_preprocessing/
│   ├── models/
│   └── utils/
├── configs/
├── notebooks/
├── outputs/
├── requirements.txt
└── README.md
```

This structure allows separate development of different modules while ensuring maintainability and scalability.

5.2 Layout Used in This Research

In this instance a more concise and focused structure was used. This was primarily due to the narrow scope of the research, specifically comparing random forest and LSTM performance on wearable sensor data, as well as the constraints of the MSc project timelines. The design was structured as follows:

```
project-root/
├── notebooks/
│   └── fall_detection.ipynb
├── data/
├── outputs/
```

└─ requirements_exact.txt

This design allowed for quicker iteration and experimentation in a more direct way (notebook-based) while still maintaining enough organisation for essential reproducibility.

6. Data Requirements

- **Primary Dataset:** SisFall (fall and non-fall activities)
- **Optional Datasets:** MHealth, WISDM (for comparison)
- **Preprocessing Steps:** Resampling, normalisation, sliding window segmentation (450 steps × 6 channels), and one-hot encoding of labels.

7. Execution Workflow

1. **Prepare Data** in data/raw/ directory.
2. **Activate Environment** using instructions in Section 4.
3. **Launch Jupyter Notebook**:

   ```
   jupyter notebook notebooks/super_fall.ipynb
   ```

4. **Run Experiments** sequentially, recording metrics and saving models to outputs/.

8. Performance Observations

- LSTM training on CPU is feasible but requires longer execution time compared to tree-based models.
- For real-time deployment at the edge, **TensorFlow Lite** with quantisation is recommended to help with inference latency and resource use.

9. Troubleshooting

- **No NVIDIA GPU Detected:** CUDA-related packages may still be installed but are unused.
- **Display Issues in Wayland:** Use inline plotting in notebooks or switch to Xorg session, whichever works better or is more feasible.
- **Memory Constraints:** If you are limited in RAM by your computer, you can reduce your batch size or window length.

10. Ethical Considerations

This configuration has the advantage of conducting sensor data processing locally, and is therefore subject to less risk given the profile nature of the data, and no personally identifiable information is being retained. However, researchers still have an obligation to adhere to the appropriate related ethics approvals for human subject testing.