National College of Ireland

Configuration Manual

MSc Research Project
Artificial Intelligence

Naveen Prasad Ravichandran
Student ID: X23415720

School of Computing
National College of Ireland

Supervisor: Dr Abdul Razzaq

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Naveen Prasad Ravichandran
Student ID:	X23415720
Programme:	Artificial Intelligence
Year:	2025
Module:	MSc Research Project
Supervisor:	Dr Abdul Razzaq
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	1664
Page Count:	10

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 December 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

Naveen Prasad Ravichandran
X23415720

1 Introduction

This manual of configuration gives the full technical setup and execution workflow needed to duplicate the hybrid physics–AI climate forecasting system developed in this research project. It describes the system requirements, libraries, dataset preparation steps, model training workflow, and evaluation procedures. The manual is intended for both technical and non-technical users, enabling anyone to rebuild the project from scratch without prior domain expertise.

2 System Requirements & Technologies Used

Table 1 summarises the key system specifications required to run the project efficiently.

Table 1: System Requirements and Technologies

Category	**Tools / Technologies**
Operating System	Windows 10/11 (Linux/MacOS also supported)
Minimum RAM	16 GB (large sequence datasets)
Storage Needed	30 GB (raw ERA5 + processed files)
Python Version	Python 3.10
Development Environment	VS Code or Jupyter
Data Processing	Pandas, NumPy, SciPy
Deep Learning Framework	PyTorch 2.x
Visualisation	Matplotlib, Seaborn
CO_2 and Climate Data Tools	ECMWF API, `eccodes`
Model Training Hardware	CPU supported; NVIDIA GPU recommended

3 Libraries and Packages Used

This project relies on a range of Python libraries that support climate data acquisition, preprocessing, model development, and evaluation. The core libraries include:

- **Pandas** – loading, cleaning, merging, and manipulating large climate datasets.
- **NumPy** – numerical computation and array-based operations.
- **Matplotlib / Seaborn** – generating analytical plots and visualising model performance.

- **PyTorch** – implementing, training, and evaluating the Baseline TCN and FiLM-TCN deep-learning models.
- **joblib** – saving and restoring trained scalers to ensure reproducibility.
- **eccodes** – decoding ERA5 GRIB-format meteorological data files.
- **tqdm** – providing progress bars during data downloads and training loops.

These libraries collectively provide the computational foundation for the full data-to-model workflow.

3.1 Install Required Python Libraries

Run inside the project folder:

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

If the file is missing, install manually:

```
pip install numpy pandas matplotlib seaborn scipy torch joblib tqdm eccodes
```

What this does: Installs all Python libraries required for data handling, modelling, and visualisation.
Why it matters: Scripts will not run unless all dependencies are installed correctly.

4 Project Workflow Overview

The climate forecasting system works in five key stages:

1. Download ERA5 raw climate data.
2. Merge and prepare the dataset.
3. Clean and finalise the dataset.
4. Train two climate forecasting models.
5. Compare their performance.

For users who prefer an automated workflow, the entire pipeline can also be executed through a single-click batch script. This optional feature is described in Section 8, which explains how `run_project.bat` orchestrates all steps—provided that the CDS API is configured correctly.

5 Dataset Source and Construction

The unified climate dataset developed for this study spans the years 1940–2100 and integrates several authoritative climate information sources. The dataset consists of:

- ERA5 historical reanalysis data (hourly meteorological variables),
- NOAA atmospheric CO_2 observations,
- SSP3–7.0 scenario-based CO_2 projections,
- synthetic trend-preserving extensions for completing the 1940–2100 horizon.

Historical meteorological variables originate from the ERA5 global reanalysis produced by the European Centre for Medium-Range Weather Forecasts Hersbach et al. (2020). Atmospheric CO_2 observations for the historical period are sourced from the NOAA Earth System Research Laboratory ESRL (2024) and complemented by the long-term emissions reconstruction presented by Hoesly et al. (2018).

Future CO_2 trajectories follow the SSP3–7.0 scenario, using CMIP6-aligned greenhouse gas concentration pathways from Meinshausen et al. Meinshausen et al. (2020) and extended using values reported in the IPCC Sixth Assessment Report I (2021). Together, these sources enable a continuous long-term climate dataset that extends beyond the observed historical range.

Raw meteorological data are initially provided in hourly GRIB format and subsequently decoded and aggregated into daily values. These records are then merged with historical and scenario-based CO_2 forcing, cleaned, interpolated, and validated for timestamp continuity. The resulting dataset supports both in-distribution (historical) and out-of-distribution (future) model evaluation under progressively increasing radiative forcing.

6 Pre-Processing and Data Preparation

This section describes each step in the data preparation pipeline, from configuring the ECMWF CDS API to downloading raw ERA5 files and producing the final cleaned dataset used for model training. Each step explains what the script does and why it is required, allowing even a non-technical user to understand the workflow.
Note : Downloading each year data may take more more than 5 hours so I suggest to skip this step 6 and proceed with step 7. raw ERA5 files are already available in

`era5_hourly_dublin_grib/`

and final cleaned Dataset

`climate_cleaned_1940_2100.csv`

are already available

6.1 Configuring the CDS API for ERA5 Downloads

To download ERA5 data programmatically, the CDS API (`cdsapi`) must be configured correctly on the user's machine. This requires:

1. Creating an account at the Copernicus Climate Data Store (CDS):
   `https://cds.climate.copernicus.eu/user/register`
2. Obtaining the API key from the user's CDS profile:
   `https://cds.climate.copernicus.eu/user/ username`
3. Creating a configuration file on the local machine:

   ```
   Windows Path:
   C:\Users\<username>\.cdsapirc

   Linux/Mac Path:
   /home/<username>/.cdsapirc
   ```

4. Copying the following contents into `.cdsapirc`:

   ```
   url: https://cds.climate.copernicus.eu/api/v2
   key: <UID>:<API-KEY>
   ```

5. Installing the CDS API Python package:

   ```
   pip install cdsapi
   ```

What this step does: It authorises your system to remotely request climate datasets from the Copernicus Climate Data Store. **Why it matters:** Without this configuration, the script `era5_downloader.py` cannot connect to the CDS servers and all downloads will fail.

6.2 Raw Data Download

The script `era5_downloader.py` downloads the core meteorological variables required for this research, including:

- 2m temperature (t2m)
- Total precipitation (tp)
- Surface solar radiation downwards (ssrd)

The script retrieves hourly ERA5 reanalysis data from 1940–2024 and stores them in:

`era5_hourly_dublin_grib/`

What this step does: It connects to the ECMWF CDS API and retrieves three essential meteorological variables used for temporal climate modelling. **Why it matters:** ERA5 is one of the most robust global reanalysis datasets Hersbach et al. (2020). These raw files form the foundation of the unified climate dataset.

6.3 Merging Raw Files

The script `dataset_raw_merge.py` loads each downloaded GRIB file, extracts the variables using `eccodes`, converts them into `pandas` dataframes, aligns timestamps, and merges all records into a single file:

`era5_dublin_1940_2024.csv`

What this step does: It consolidates hundreds of yearly GRIB files into one unified CSV file by aggregating hourly measurements into daily statistics. **Why it matters:** A single consolidated dataset is essential for downstream cleaning, CO_2 merging, and feature engineering. Machine learning models cannot work directly with GRIB files.

6.4 Adding CO_2 and Temporal Features

The script `datapreparation.py` enriches the merged ERA5 data by adding external forcing and temporal descriptors. It performs:

- Integration of historical atmospheric CO_2 observations (1940–2020) from NOAA ESRL ESRL (2024) and Hoesly et al. (2018).
- Addition of SSP3–7.0 CO_2 scenario projections (2025–2100) from Meinshausen et al. Meinshausen et al. (2020) and IPCC AR6 I (2021).
- Generation of extended daily rows up to the year 2100.
- Computation of temporal features: year, month, day, season.
- Saving of feature scaling files for model training.

This step produces:

`climate_1940_2100.csv`

What this step does: It transforms the meteorological dataset into a forcing-aware dataset by explicitly adding CO_2 concentrations and temporal encodings. **Why it matters:** The research question investigates whether CO_2 forcing improves extrapolation stability. Without this step, the FiLM-TCN cannot learn forcing–response relationships.

6.5 Cleaning the Dataset

The script `dataset_cleaning.py` performs the final preparation, ensuring the dataset is ready for sliding-window sequence construction. It:

- Verifies that all timestamps form a continuous 30,474-day sequence.
- Fills missing values using interpolation.
- Standardises column names and formatting.

It outputs:

`climate_cleaned_1940_2100.csv`

What this step does: Ensures dataset completeness, consistency, and correct formatting for deep-learning models. **Why it matters:** Temporal models require continuous sequences. Missing dates or corrupted values would break the training pipeline and cause unstable predictions.

6.6 EDA Analysis

The script `EDA.py` performs Complete EDA analysis. outputs available in :

`EDA_Analysis`

What this step does: Checks dataset completeness, verifies continuity of the time index, and confirms that all variables follow a consistent format required by the model. It also identifies missing timestamps, gaps, or corrupted entries that could disrupt downstream processing. **Why it matters:** Deep-learning sequence models depend on uninterrupted, correctly ordered time-series data. Even a single missing timestamp or malformed value can break sliding-window sequence construction, distort feature engineering, and lead to unstable or physically unrealistic predictions. Ensuring clean, continuous data is therefore essential for reliable climate forecasting.

7 Model Training and Configuration

This section describes how both climate forecasting models—the Baseline TCN and the FiLM–TCN—are trained, configured, and evaluated. Each step includes a clear explanation of what the process does and why it is necessary. All training scripts use the final cleaned dataset `climate_cleaned_1940_2100.csv` and the feature engineering pipeline defined earlier.

7.1 Training the Baseline TCN Model

To train the forcing-agnostic Temporal Convolutional Network, run:

```
python baseline_TCN_training.py
```

This produces the trained model file:

`baseline_tcn_v3_best.pth`

What this step does: The script trains a standard Temporal Convolutional Network that receives only meteorological and temporal features. It learns patterns such as temperature seasonality, precipitation variability, and short–term climate fluctuations.

Why it matters: This model serves as the "control" architecture. Because it does not receive CO_2 forcing information, its performance provides a baseline for assessing whether explicit forcing improves long–range prediction robustness.

7.2 Training the FiLM–TCN (Forcing–Aware) Model

To train the CO_2–aware model, run:

```
python Film_TCN_trainingv13.py
```

This produces:

`film_tcn_v13_best.pth`

What this step does: This model uses Feature–wise Linear Modulation (FiLM) to incorporate CO_2 forcing. A secondary network converts CO_2 descriptors into modulation parameters (γ, β), which adjust the internal feature maps of the network.

Why it matters: This step directly tests the research question: whether adding explicit CO_2 forcing improves predictive stability under strong future forcing scenarios.

7.3 Model Evaluation

To evaluate the baseline model:

```
python baseline_evaluate.py
```

To evaluate the FiLM–TCN:

```
python film_tcn_evaluate.py
```

Outputs include:

- RMSE, MAE, and R^2 performance scores,
- Prediction vs. ground-truth plots,
- Residual analysis visualisations.

What this step does: This stage measures how well each model predicts temperature and precipitation across both in–distribution (1940–2080) and out–of–distribution (2081–2100) periods.

Why it matters: It provides direct evidence of whether forcing-aware conditioning reduces error growth under high CO_2 climate regimes.

7.4 Model Comparison

To compare the two models directly, run:

```
python compare_models.py
```

All outputs are saved in:

```
comparison_outputs/
```

Includes:

- Metric comparison bar charts,
- CO_2-dependent residual plots (ID and OOD),
- Time-series predictions for both models under historical and future climate conditions.

What this step does: This script compiles all evaluation components into a single visual comparison suite.

Why it matters: These visual summaries clearly show whether the FiLM–TCN is more stable and accurate than the Baseline TCN under extreme forcing conditions, allowing the research conclusions to be drawn objectively.

7.5 statistical analysis

To run the statistical analysis, run:

```
python stats.py
```

All outputs are saved in:

```
statistical_analysis/
```

8 Optional Single-Click Execution Using `run_project.bat`

To simplify the workflow, the entire project can be executed using a single batch script:

`run_project.bat`

This script automates the complete pipeline—from downloading ERA5 data to training both models and generating comparison outputs. Users do not need to run individual scripts manually. However, the batch execution requires that the CDS API be configured correctly on the system.
Note : `Datapreparation` steps is Commented by default because all ERA5 files are already available in

`era5_hourly_dublin_grib/`

so i suggest to skip this time consuming step8.1 and final cleaned Dataset

`climate_cleaned_1940_2100.csv`

are already available

8.1 CDS API Configuration Requirement

The batch script relies on `era5_downloader.py`, which retrieves ERA5 reanalysis data from the Copernicus Climate Data Store (CDS). For this to work, the CDS API must be fully configured. The setup process includes the following steps:

1. Create a Copernicus CDS account:
   `https://cds.climate.copernicus.eu/user/register`
2. Locate the personal API key from the CDS user profile:
   `https://cds.climate.copernicus.eu/user/`
3. Create a configuration file named `.cdsapirc` in the user directory.
   For Windows:

   ```
   C:\Users\<username>\.cdsapirc
   ```

   For Linux/Mac:

   ```
   /home/<username>/.cdsapirc
   ```

4. Add the following contents to the file:

   ```
   url: https://cds.climate.copernicus.eu/api/v2
   key: <UID>:<API-KEY>
   ```

5. Install the CDS API package:

   ```
   pip install cdsapi
   ```

Why this matters: Without proper CDS API configuration, the ERA5 download step will fail, causing the batch script to terminate early.

8.2 What `run_project.bat` Does

Once the CDS API is configured, the batch file automates the following sequence:

1. Downloads ERA5 historical climate data.
2. Merges raw GRIB files into a unified CSV dataset.
3. Adds CO_2 forcing and temporal features.
4. Cleans the dataset and prepares it for modelling.
5. Trains the Baseline TCN model.
6. Trains the FiLM–TCN forcing-aware model.
7. Evaluates each model separately.
8. Generates comparative visualisations for ID and OOD regimes.
9. Perform Statistical analysis.
10. Saves all results into structured output folders.

Why this is useful: This single-click workflow ensures that all steps are executed correctly and in order, making the project easy to reproduce even for non-technical users.

8.3 How to Run the Batch File

Place `run_project.bat` in the project directory (`CLIMATE-V3`) and execute it by:

`Double-clicking run_project.bat`

or running from the terminal:

```
./run_project.bat
```

8.4 Troubleshooting

- **ERA5 download fails:** Ensure the `.cdsapirc` file is correctly placed and contains a valid API key.
- **"cdsapi not found":** Install the package using `pip install cdsapi`.
- **Memory errors:** Close background applications or reduce the training batch size.
- **Slow execution:** The workflow runs on CPU if no GPU is available; this is expected to be slower.

This optional batch execution method provides a streamlined, repeatable process for running the entire project pipeline with a single command.

References

ESRL, N. (2024). Climate monitoring & diagnostics laboratory – co_2 global monitoring. Accessed 2024.
URL: *https://gml.noaa.gov/ccgg/trends/*

Hersbach, H., Bell, B., Berrisford, P., et al. (2020). The era5 global reanalysis, *Quarterly Journal of the Royal Meteorological Society* **146**(730): 1999–2049.
URL: *https://rmets.onlinelibrary.wiley.com/doi/10.1002/qj.3803*

Hoesly, R. M., Smith, S. J., Feng, L. et al. (2018). Historical (1750–2014) anthropogenic emissions in the shared socioeconomic pathways, *Geoscientific Model Development* **11**: 369–408.

I, I. W. G. (2021). Climate change 2021: The physical science basis (chapter 4; table 4.sm.2).
URL: *https://www.ipcc.ch/report/ar6/wg1/*

Meinshausen, M. et al. (2020). The ssp greenhouse gas concentrations and their impacts on radiative forcing based on cmip6, *Geoscientific Model Development* **13**: 3571–3605.
URL: *https://doi.org/10.5194/gmd-13-3571-2020*