National College of Ireland

Configuration Manual

MSc Research Project
Artificial Intelligence

Muhammad Anis Ur Rahman
Student ID: 23284803

School of Computing
National College of Ireland

Supervisor: Abdul Shahid

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Muhammad Anis Ur Rahman
Student ID:	23284803
Programme:	Artificial Intelligence
Year:	2025
Module:	MSc Research Project
Supervisor:	Abdul Shahid
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	654
Page Count:	5

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:	15th September 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

Muhammad Anis Ur Rahman
23284803

1 Introduction

This installation guide covers the installation, configuration, and execution of the project built for the research project MSc *Evaluating the Impact of Data Augmentation on Image Classification Accuracy*. The project enables reproducible exploration of vying benchmarking data augmentation schemes of differing convolutional and new deep learning architectures on standard benchmark datasets like CIFAR-10, MNIST, and ImageNet.

It comes on Windows 10/11 (64-bit), macOS (Monterey and later), and Linux (Ubuntu 20.04/22.04 LTS). Execution of code on CPU and GPU and GPU acceleration are supported and advised if you train on big models or computationally demanding data augmentations. For GPU runs, an NVIDIA GPU with compute capability ≥ 6.0 and at least 6 GB of VRAM and CUDA 11.8 or 12.1 and cuDNN 8.5+ is advised. CPU-only runs on the other hand are supported but slower-running training of deep models and big data sets.

2 Environment Requirements

The project is backed by a number of operating systems and hardware configurations. Specs that are minimum and optimal are presented in Table 1.

Table 1: Minimum and recommended environment specifications

Component	**Minimum**	**Recommended**
CPU	4-core x64 processor	8+ cores
RAM	8 GB	16 GB or more
Disk Space	20 GB free	50 GB
GPU	Optional	6 GB+ VRAM, Compute ≥ 6.0
Python	3.8+	3.10 or 3.11
CUDA/cuDNN	N/A (CPU)	CUDA 11.8 or 12.1, cuDNN 8.5+
OS	Win 10/11, macOS 12+, Ubuntu 20.x	

3 Installation

1. Clone the repository

```
git clone https://github.com/iAnisDev/data-augmentation-impact.git
cd data-augmentation-impact
```

2. Create and activate a Python environment

Using `venv`:

```
python -m venv venv
# macOS/Linux:
source venv/bin/activate
# Windows:
venv\Scripts\activate
```

Or with Conda:

```
conda create -n augenv python=3.10 -y
conda activate augenv
```

3. Install dependencies

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

Linux prerequisites.

```
sudo apt update && sudo apt install -y python3-dev build-essential
```

Windows prerequisites. Ensure the *Visual C++ Build Tools* are installed so required Python packages can compile successfully.

4 Datasets

The project supports manual and automatic setting of up datasets. Automatic downloading of CIFAR-10 and MNIST is supported while ImageNet is downloaded manually due to licensing. All the datasets ultimately get stored inside the `.data/` directory in the project root. Default auto-downloaded data will be saved under their data name with distinct `train/` and `test/` directories

1. Automatic download

Use the `--load-data` flag to download a supported dataset:

```
python src/main.py --load-data --dataset cifar10
```

Supported values for `--dataset` are:

```
cifar10
mnist
imagenet
all
```

2. Raw dataset structure

Automatically downloaded datasets are saved under their dataset name with separate `train/` and `test/` folders:

```
.data/
  cifar10/
    train/
      class1/
      class2/
      ...
    test/
      class1/
      class2/
      ...
```

3. Processed and augmented structure

After preprocessing and data enhancement, the preprocessed data sets are stored under:

```
.data/
  processed/
    cifar10/
      test/
      train/
        auto/
        fusion/
        gan/
        lsb/
        miamix/
        mixup/
        traditional/
        vqvae/
```

The `test/` directory remains as is and `train/` contains one subdirectory per data augmentation method. In this way, multiple data augmentation methods may coexist without overwriting each other.

5 Running the Framework

The project is operated through a extended command-line interface, allowing you to run the complete pipeline or individual stages.

1. Full pipeline

Run all stages (download, preprocess, train, evaluate) together in a single command:

```
python src/main.py --all --dataset cifar10 --model resnet18 --augment traditional
```

2. Individual stages

Each stage can be executed separately:

```
# Download data
python src/main.py --load-data --dataset cifar10

# Preprocess and augment data
python src/main.py --preprocess --dataset cifar10 --augment gan

# Train model
python src/main.py --train --dataset cifar10 --model resnet18 --augment gan

# Evaluate model
python src/main.py --evaluate --dataset cifar10 --model resnet18 --augment gan
```

3. Using a configuration file

Experiments can also be run from a YAML or JSON configuration file:

```
python src/main.py --all --config config.yaml
```

Values in the configuration file will override command-line defaults, ensuring experiments can be reproduced exactly for validation purposes.

6 Outputs and Logs

After training and evaluation, all results are stored in well-defined directories to maintain reproducibility and validation.

1. Processed datasets

Processed and augmented datasets are stored in:

```
.data/processed/<dataset>/
  test/
  train/<augmentation>/
```

The `test/` folder remains unchanged, while `train/` contains one folder per augmentation method.

2. Model weights

Each trained model is saved under:

```
weights/<model>_<dataset>/model_<augmentation>.pt
```

This naming convention ensures that multiple experiments versions can be stored without overwriting previous results.

3. Training and evaluation logs

Logs and metrics are stored in `.artifacts/`:

```
.artifacts/train_log_<model>_<dataset>_<augmentation>.json
.artifacts/eval_report_<model>_<dataset>_<augmentation>.json
.artifacts/combined_results_<dataset>_<augmentation>.json
```

These involve precision, loss curves, training time, confusion matrix, and other measures of performance.

Appendix: Ready-to-Run Configuration File

The following configuration file can be taken as an example of training one full train and test cycle on CIFAR-10 using ResNet-18 and standard augmentations. Store this content as `config.yaml` in the project root.

```
dataset: "cifar10"
model: "resnet18"
augmentation: "traditional"
epochs: 20
batch_size: 64
pretrained: false
device: "auto"  # auto-detect GPU or CPU

# Advanced settings
learning_rate: 1e-3
weight_decay: 1e-4
scheduler: "cosine"
```

Run the experiment using:

```
python src/main.py --all --config config.yaml
```

This setting offers a reproducible starting point of an experiment. You can reproduce and scale up for other data sets, other data augmentations, or other model architectures while you hold everything else constant.