National College of Ireland

# Configuration Manual

MSc Research Practicum Part 2
MSc in Data Analytics

Joseph Jacob Anjilimoottil
Student ID: 24103390

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Joseph Jacob Anjilimoottil |
| **Student ID:** | 24103390 |
| **Programme:** | MSc in Data Analytics (MSCDAD_A_JAN25I) |
| **Programme:** | MSc in Data Analytics (MSCDAD_A_JAN25I) |
| **Module:** | MSc Research Practicum Part 2 |
| **Supervisor:** | Vikas Sahni |
| **Submission Due Date:** | 11th December 2025 |
| **Project Title:** | From Binary to Multiclass: Adapting the IT Transformer for Fine-Grained Alzheimer's Disease Stage Classification |
| **Word Count:** | 899 **Page Count**: 8 |

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:** Joseph Jacob Anjilimoottil

**Date:** 10 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

Joseph Jacob Anjilimoottil
Student ID: 24103390

# 1 Overview

This Configuration Manual presents all system parameters, software requirement, dataset preparation guidelines and model configuration parameters to replicate the multiclass Alzheimer disease classification system that was developed during the project. It was done in Python and PyTorch and similar machine-learning tools and ran in Google Colab Pro, which uses the A100 as the GPU accelerator.

This document guarantees that the environment can be set up by another user; data prepared, models configured and the training and evaluations processes run without the original notebook being accessed.

# 2 Hardware Used

| **Component** | **Hardware Used** |
|---|---|
| GPU | NVIDIA A100 |
| CPU | Quad-core processor or higher |
| RAM | 12 GB |
| Disk Space | 20 GB (for dataset + models + outputs) |
| Environment | Google Colab Pro |
| Operating System | Ubuntu 22.04 |

# 3 Software Used

## 3.1 Programming Language

- Python 3.12

## 3.2 Machine Learning Libraries

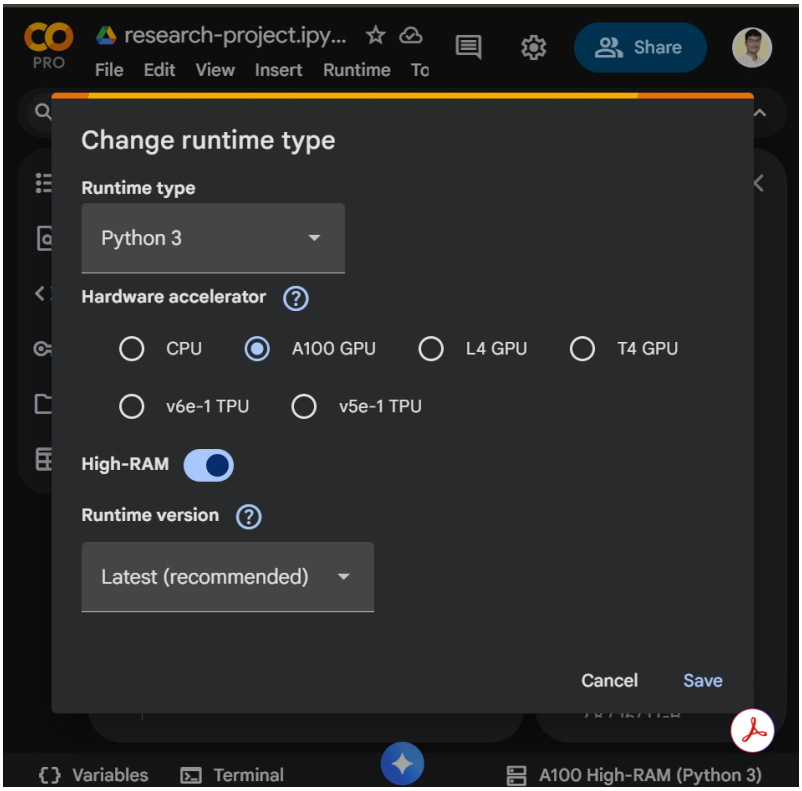
| **Library** | **Version** |
|---|---|
| torch | 2.9.0+cu126 |
| torchvision | 0.24.0+cu126 |
| seaborn | 0.13.2 |
| matplotlib | 3.10.0 |
| kagglehub | 0.3.13 |
| sklearn | 1.6.1 |

| | |
|---|---|
| pandas | 2.2.2 |
| PIL | 11.3.0 |
| numpy | 2.0.2 |
| timm | 1.0.22 |

# 4 Environment Setup

## 4.1 GPU Runtime Configuration (Colab)

1. Open Change Runtime Type
2. Set:
   - Hardware accelerator: GPU
   - GPU type: A100 (Pro required)
   - Runtime: Python 3




## 4.2 Install Required Libraries

All the libraries needed in this research have been pre-installed in Colab.In case not, use this in Colab cell:

```
pip install torch torchvision
pip install scikit-learn
pip install matplotlib seaborn
```

### 4.3 Verify GPU Availability

```
import torch
torch.cuda.is_available()
torch.cuda.get_device_name(0)
```

Output should show:

*True*
*'NVIDIA A100-SXM4-40G*

# 5 Dataset Configuration

## 5.1 Dataset Used

Alzheimer Brain MRI  Dataset

Directory structure:

```
combined_images/
├── NonDemented
├── VeryMildDemented
├── MildDemented
└── ModerateDemented
```

```
from torchvision import datasets
import torchvision.transforms as transforms
from torch.utils.data import DataLoader, Subset
from sklearn.model_selection import train_test_split
import os
# /kaggle/input/alzheimers-multiclass-dataset-equal-and-augmented/combined_images
DATA_DIR = "/kaggle/input/alzheimers-multiclass-dataset-equal-and-augmented/combined_images"

# Load raw dataset WITHOUT transforms (needed for EDA)
raw_dataset = datasets.ImageFolder(DATA_DIR)

print(" Dataset loaded")
print("Classes:", raw_dataset.classes)
print("Total images:", len(raw_dataset))
```

```
 Dataset loaded
Classes: ['MildDemented', 'ModerateDemented', 'NonDemented', 'VeryMildDemented']
Total images: 44000
```

## 5.2 Uploading / Linking Dataset

Two options:

**Option A — Using Kaggle API in Colab**

1. Upload kaggle.json to Colab
2. Run:

```
!mkdir ~/.kaggle
```

```
!mv kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -daryansinghal10/alzheimers-multiclass-dataset-equal-and-augmented
!unzip alzheimers-multiclass-dataset-equal-and-augmented.zip
```

**Option B — Upload folder manually**

Upload entire dataset to:

```
/content/combined_images
```

## 5.3 Data Transformations

The preprocessing of all images was done with:

- Resize to 224 × 224
- Normalization: mean = 0.5, std = 0.5
- Augmentations:
  - Random rotation
  - Random horizontal flip
  - Random affine
  - Color jitter
  - Gaussian blur
  - Random cropping

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.RandomResizedCrop(224, scale=(0.9, 1.0)),
    transforms.ColorJitter(brightness=0.2, contrast=0.2),
    transforms.ToTensor(),
    transforms.Normalize([0.5], [0.5]),
    transforms.RandomAffine(degrees=10, translate=(0.05, 0.05), scale=(0.95, 1.05)),
    transforms.GaussianBlur(kernel_size=(3,3), sigma=(0.1, 1.0)),

])

# Load full dataset
full_dataset = datasets.ImageFolder(DATA_DIR, transform=transform)
```

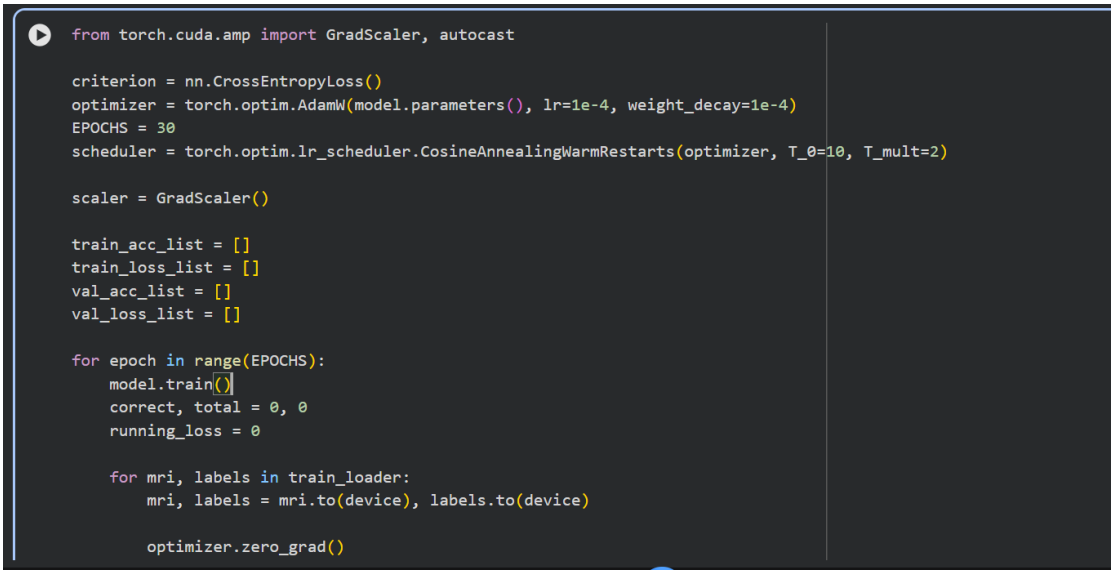
These changes should be continuously used in the pre-training stage.

# 6 Model Configuration

## 6.1 Interpretable Transformer

**Model Hyperparameters**

| Parameter | Value |
|---|---|
| Learning Rate | 1e-4 |
| Optimizer | AdamW |
| Scheduler | Cosine Annealing |
| Loss | CrossEntropyLoss |
| Epochs | 30 |
| Batch Size | 32 |
| Mixed Precision | Enabled (AMP) |



```
from torch.cuda.amp import GradScaler, autocast

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-4, weight_decay=1e-4)
EPOCHS = 30
scheduler = torch.optim.lr_scheduler.CosineAnnealingWarmRestarts(optimizer, T_0=10, T_mult=2)

scaler = GradScaler()

train_acc_list = []
train_loss_list = []
val_acc_list = []
val_loss_list = []

for epoch in range(EPOCHS):
    model.train()
    correct, total = 0, 0
    running_loss = 0

    for mri, labels in train_loader:
        mri, labels = mri.to(device), labels.to(device)

        optimizer.zero_grad()
```

# 7 Training Configuration

## 7.1 Training Pipeline

1. Load augmented dataset
2. Apply stratified 70/10/20 split
3. Instantiate the choice of model (IT or CNN)
4. Move model to GPU
5. Start mixed precision
6. Train for 30 epochs
7. Save checkpoint with the best performance.

## 7.2 Training Outputs

- Training accuracy curve
- Training loss curve
- Validation accuracy & loss
- Best model .pth file
- Scheduler & optimizer states

These are required outputs needed in reporting and reproducibility.

# 8 Evaluation Configuration

## 8.1 Evaluation Metrics

- Test accuracy
- Per-class precision
- Per-class recall
- Per-class F1-score
- Confusion matrix

These metrics were computed with the help of Scikit-learn.

## 8.2 Required Evaluation Steps

1. Load trained model checkpoint
2. Disable gradient tracking
3. Run inference on test set
4. Generate confusion matrix
5. Output classification report.
6. Log final accuracy

## 8.3 Evaluation Outputs

- Confusion matrix heatmap
- Report table on classification.
- Accuracy comparison chart

# 9 Reproducibility Settings

To achieve consistent outputs:

| Setting | Value |
|---|---|
| Random seed | 42 |
| Num workers | 2 |
| Device | GPU-only (no CPU fallback) |

# 10 Summary

This configuration manual contains all the settings, environment description, model specifications, dataset preparation procedures and evaluation steps necessary to recreate the Alzheimer multiclass MRI classification system. Through these configurations, one researcher may reproduce the same experimental setting and get results that are comparable to those of the original project.