National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

UNNIKRISHNAKURUP AMBIKA REGHUNATHA KURUP
Student ID: 23334011

School of Computing
National College of Ireland

Supervisor: Athanasios Staikopoulos

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	UNNIKRISHNAKURUP AMBIKA REGHUNATHA KURUP
Student ID:	23334011
Programme:	Master of Science in Data Analytics (MSCDAD_A_JAN25I) **Year:** 2026
Module:	Research Practicum Part 2
Lecturer:	Athanasios Staikopoulos
Submission Due Date:	28/01/2026
Project Title:	Multi-Level Attention Graph Network for Enhanced Depression Type Detection
Word Count:	**907** **Page Count: 11**

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: 27/01/2026

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

UNNIKRISHNAKURUP AMBIKA REGHUNATHA KURUP
Student ID: 23334011

1 Introduction

This configuration file contains the complete details needed to recreate the classification system for the subtype of depression 'MAGNET-D'. The configuration file encompasses environment settings, versions of the libraries, data preparation, architectural parameters for the model, and the training procedures to reach an accuracy of 86.22% in six-class identification of depression present in the Twitter posts.

Objective: Replicating results precisely in any experiment
Environment: Google Colab Pro with GPU acceleration
Dataset: Multi-Class Depression Detection Dataset (14,983 tweets) https://zenodo.org/records/14233292

2 Environment Setup

2.1 Platform

Environment: Google Colab Pro
GPU: NVIDIA Tesla T4 (16GB VRAM)
RAM: System RAM ~25GB
Storage: Google Drive mounted for dataset access

Python Version

Python: 3.10.x (Colab default)

System Configuration

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
```

2.2 Local Machine

Component	Specification
Device Name	LAPTOP-VHBTKCIO
CPU	AMD Ryzen 7 7730U with Radeon Graphics (2.00 GHz)
RAM	16 GB

Component	Specification
OS	Windows 11 Home Single Language, Version 24H2 (OS Build 10.0.26100 Build 26100)
System Type	64-bit operating system, x64-based processor

3 Library Requirements

Installation Commands:

```
!pip install -q torch torchvision torchaudio
!pip install -q torch-geometric
!pip install -q sentence-transformers wordcloud scikit-learn
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.7/63.7 kB 3.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 44.1 MB/s eta 0:00:00
```

Key Library Versions

PyTorch: 2.x
PyTorch Geometric: Latest compatible with PyTorch 2.x
sentence-transformers: Latest (tested with 2.2+)
scikit-learn: Latest (tested with 1.3+)
numpy: 1.23+
pandas: 1.5+
matplotlib: 3.7+
seaborn: 0.12+

Import Structure

```
import re
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from wordcloud import WordCloud
from sentence_transformers import SentenceTransformer

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    f1_score,
    accuracy_score
)
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch_geometric.data import Data
from torch_geometric.nn import GATv2Conv
from sklearn.neighbors import kneighbors_graph
```

4 Dataset Requirements

Dataset Source

- Name: Multi-Class Depression Detection Dataset
- Source: Zenodo (https://zenodo.org/records/14233292)
- Format: CSV file

Dataset Specifications

- Total samples: 14,983 tweets
- Features: clean_text, label, word_len (word count)
- Target classes: 6 depression subtypes
- Major Depressive Disorder
- Bipolar Depression
- Psychotic Depression
- Atypical Depression
- Postpartum Depression
- No Depression

Class Distribution

- Class imbalance present (moderate)
- Stratified sampling used for train/test split

Data Loading

```
path = "dataset.csv"
df = pd.read_csv(path)
print("Raw shape:", df.shape)
print(df.head())
print("Raw columns:", df.columns.tolist())
```

5 Project Directory

```
MAGNET_D/
├── dataset.csv              # Raw dataset
├── dataset_clean.csv        # Preprocessed dataset
├── MAGNET_D_23334011.ipynb  # Main notebook
└── Configuration Manual     # Project documentation
```

6 Graph Construction Configuration

Graph Type

- Structure: K-Nearest Neighbors (KNN) graph
- Node representation: Tweet embeddings (384-dim)
- Edge definition: Semantic similarity between tweets

KNN Parameters

```
adj = kneighbors_graph(
    embeddings,
    n_neighbors=8,
    mode="connectivity",
    include_self=False
)
```

Graph Statistics

- Nodes: 14,983 (one per tweet)
- Edges per node: 8
- Total edges: ~119,864 directed edges
- Graph type: Undirected, unweighted

Embedding Generation

- Model: all-MiniLM-L6-v2 (sentence-transformers)
- Output dimension: 384
- Normalization: L2-normalized for cosine similarity
- Batch size: 128 (for encoding)

7 MAGNET-D Model Configuration

Architecture Components

- Feature-Aware Edge Updater (FAEU)
- Symptom-Aware GATv2
- Cross-Level Attention (CLA)
- Multi-Task Learning Head (MTL)
- Symptom Prototypes
- Full Model Instantiation

Model Size

- Total parameters: 242,000 trainable parameters
- Memory footprint: 1 MB (model weights)

```python
class MAGNETD_Full(nn.Module):
    def __init__(self, embed_dim, hidden_dim, symptom_vectors, user_dim, num_classes=6):
        super(MAGNETD_Full, self).__init__()

        self.faeu = FAEU(
            symptom_vectors=symptom_vectors,
            embed_dim=embed_dim,
            hidden_dim=hidden_dim
        )

        self.sym_gat = SymptomAwareGAT(
            in_dim=embed_dim,
            hidden_dim=hidden_dim,
            symptom_vectors=symptom_vectors
        )

        self.cla = CrossLevelAttention(
            tweet_dim=hidden_dim * len(symptom_vectors),
            symptom_dim=embed_dim,
            user_dim=user_dim,
            out_dim=256
        )

        self.mtl = MAGNETD_MTL_Head(
            in_dim=256,
            num_classes=num_classes
        )
```

8 Model Configuration (MAGNET-D)

8.1 Dataset Split

- Train: 80% (11,986 samples)
- Test: 20% (2,997 samples)
- Stratified split to preserve class ratio

8.2 Optimization Strategy

- Optimizer: AdamW
- Learning Rate: 0.001
- Weight Decay: 0.0001
- Epochs: 100
- Batch Mode: Full graph training

8.3 Loss Functions

- Main Task: Cross-Entropy Loss
- Auxiliary Tasks:
- Regression: MSE Loss
- Emotion: Cross-Entropy
- Cluster: Cross-Entropy

8.4 Loss Weighting

- Intensity: 0.4
- Emotion: 0.2
- Cluster: 0.2

8.5 Regularization

- Dropout: 0.3
- L2 Regularization

9 Training Configuration

Training Hyperparameters

- Epochs: 100
- Batch mode: Full-batch
- Early stopping: Enabled (patience = 10)
- Gradient clipping: Not used

Regularization Techniques

- Dropout: 0.3 (after each GATv2 layer)
- Weight decay: 1e-4 (L2 regularization)
- L2 normalization: Applied to embeddings before graph construction
-

Training Loop

```
from sklearn.metrics import confusion_matrix, f1_score
import numpy as np

loss_history = []
acc_history = []

optimizer = torch.optim.Adam(model.parameters(), lr=1e-3, weight_decay=1e-4)

lambda_tuple = (0.4, 0.2, 0.2)
EPOCHS = 100
print("\nStarting training...\n")

for epoch in range(1, EPOCHS + 1):

    labels_tuple = (lbl_main, lbl_intensity, lbl_emotion, lbl_cluster)

    loss = train_magnetd(
        model,
        data_full,
        symptom_matrix,
        user_features,
        labels_tuple,
        optimizer,
        lambda_tuple
    )

    loss_history.append(loss)

    if epoch % 2 == 0:
        preds = evaluate_magnetd(model, data_full, symptom_matrix, user_features)
        acc = (preds[test_mask.cpu().numpy()] == labels[test_mask.cpu().numpy()]).mean()
        acc_history.append(acc)
        print(f"Epoch {epoch:02d} | Loss={loss:.4f} | TestAcc={acc:.4f}")
```

10 Evaluation Configuration

Metrics

```python
def evaluate_magnetd(model, data, symptom_matrix, user_features):
    model.eval()
    with torch.no_grad():
        outputs = model(data, symptom_matrix, user_features)
        out_main, _, _, _ = outputs
        preds = out_main.argmax(dim=1).cpu().numpy()
    return preds
```

Primary Metrics

- **Accuracy**: Overall classification accuracy
- **Macro-F1**: Unweighted average F1 across all classes

Per-Class Metrics

- Precision, Recall, F1-score for each of 6 classes
- Support (number of samples per class)

Evaluation Protocol

- **Test set**: Hold-out 20% (never seen during training)
- **No cross-validation**: Single train/test split
- **Stratified split**: Maintains class distribution

Confusion Matrix

- **Size**: 6×6 (for 6 classes)
- **Visualization**: Heatmap with annotations

Baseline Comparisons

- **Multinomial Naive Bayes** (TF-IDF features)
- **Linear SVM** (PCA-reduced embeddings)
- **Logistic Regression** (Full embeddings)
- **Simplified Heterogeneous GNN**

11 Reproducibility Notes

Random Seed Settings

- All seeds set to 42 (Python, NumPy, PyTorch)
- Deterministic flags enabled for CUDA

```
from wordcloud import WordCloud
from sentence_transformers import SentenceTransformer

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    f1_score,
    accuracy_score
)
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch_geometric.data import Data
from torch_geometric.nn import GATv2Conv
from sklearn.neighbors import kneighbors_graph

sns.set(style="whitegrid")

SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
```

Environment Stability

- Results stable on Tesla T4 GPU
- Slight differences possible due to GPU non-determinism

Checklist for Accurate Reproduction

- Use Python 3.10.x
- Use Colab Pro GPU
- Ensure all embeddings regenerated fresh
- Verify graph structure matches original

Expected Variations

- Accuracy drift up to +/- 0.3%
- Macro-F1 drift up to +/-0.005

References

Brody, S., Alon, U., & Yahav, E. (2022). How attentive are graph attention networks? In International Conference on Learning Representations.

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95.

McKinney, W. (2010). Data structures for statistical computing in Python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51-56).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (pp. 8024-8035).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.