National College of Ireland

Configuration Manual

MSc Research Project
PMsc in Data Analytics

Srihitha Masireddy
Student ID: x23325275

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Srihitha Masireddy
Student ID:	X23325275
Programme:	Msc in Data Analytics **Year:**2025
Module:	Research Practicum
Lecturer:	Vladimir Milosavljevic
Submission Due Date:	15/09/25
Project Title:	Enchancing Phishing Email Detection Through Multi Modal Deep Learning
Word Count:	1678 **Page Count:** 12

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: Masireddy Srihitha

Date: 14-09-25

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

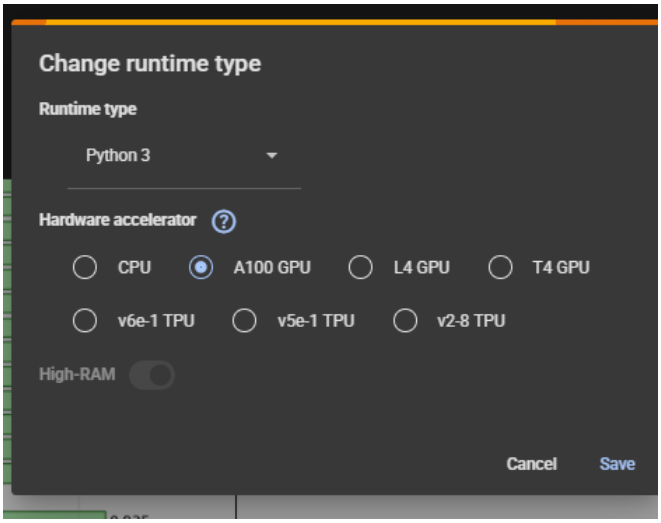
Srihitha Masireddy
X23325275

1 Google Colab Environment Setup

1.1 Initial Configuration

This implementation requires Google Colab for optimal performance due to memory requirements. Upon starting a new notebook, ensure GPU acceleration is enabled:

1. Navigate to Runtime → Change runtime type
2. Select GPU as Hardware accelerator (preferably T4 or V100)
3. Select High-RAM if available in your subscription



1.1.1 Essential Package Installation

Begin by installing all required packages. Run the following commands in order:
bash
 # Core deep learning frameworks
!pip install torch>=1.9.0 transformers>=4.0.0 -q
!pip install torch-geometric torch-scatter torch-sparse -q

 # Data processing and utilities
!pip install pandas numpy scikit-learn matplotlib seaborn tqdm -q
!pip install nltk spacy -q

 # Download NLTK data
import nltk
nltk.download('punkt')
nltk.download('stopwords')

```
nltk.download('wordnet')
nltk.download('punkt_tab')
```

1.1.2 GPU Verification

Verify GPU availability and specifications:

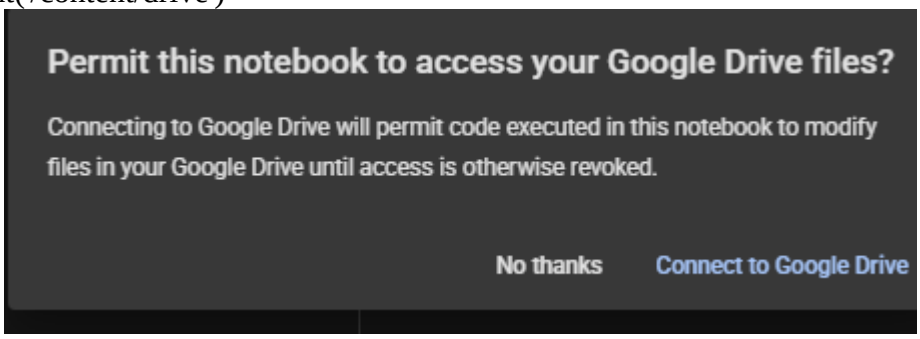
```
import torch
print(f"PyTorch version: {torch.__version__}")
print(f"CUDA available: {torch.cuda.is_available()}")
if torch.cuda.is_available():
    print(f"Current GPU: {torch.cuda.get_device_name()}")
    print(f"GPU Memory: {torch.cuda.get_device_properties(0).total_memory / 1e9:.2f} GB")
```

2 Dataset Configuration

2.1 Mounting Google Drive

If your dataset is stored in Google Drive:

```
from google.colab import drive
drive.mount('/content/drive')
```




```
# Navigate to your dataset directory
import os
os.chdir('/content/drive/MyDrive/phishing-detection/')
```

3 Loading the Phishing Email Dataset

The system expects the dataset in CSV format with columns: 'Email Text' and 'Email Type'. Load and verify the dataset:

```
import pandas as pd
# Load dataset
df = pd.read_csv('Phishing_Email.csv')
print(f"Dataset shape: {df.shape}")
print(f"Columns: {df.columns.tolist()}")
print(f"\nClass distribution:")
print(df['Email Type'].value_counts())
```

4 Creating Data Directories

Set up the required directory structure:

```
import os
directories = ['prepared_data', 'phase1_models', 'phase1_outputs',
               'phase2_models', 'phase2_outputs', 'phase3_models',
               'phase3_outputs', 'phase4_outputs']
```

for dir in directories:
os.makedirs(dir, exist_ok=True)

5 Phase-wise Execution

5.1 Phase 0: Data Preprocessing

Run the preprocessing pipeline to prepare the dataset:

 # The preprocessing code expects the dataset loaded as 'df'
 # It will create train/val/test splits and save them

! preprocessing.py

 # Or run the preprocessing notebook cells
 # This creates prepared_data/.csv files*

5.2 Phase 1: XLNet Semantic Analysis

Configure and run XLNet training with memory optimization:

 # Memory optimization for Google Colab
import gc
torch.cuda.empty_cache()
gc.collect()

 # Configure batch size based on available GPU
 # A100 (40GB): batch_size = 16
 # V100 (16GB): batch_size = 8
 # T4 (16GB): batch_size = 4

BATCH_SIZE = 8 *# Adjust based on your GPU*
MAX_LENGTH = 512 *# Can reduce to 256 if memory issues persist*
EPOCHS = 5 *# Reduce if training takes too long*

 # Run Phase 1
! xlnet_training.py --batch_size $BATCH_SIZE --max_length $MAX_LENGTH --epochs $EPOCHS

5.3 Phase 2: GNN Structural Analysis

Configure graph construction parameters:

 # GNN can handle larger batches
GNN_BATCH_SIZE = 32
GNN_EPOCHS = 15

 # Run Phase 2
! gnn_training.py --batch_size $GNN_BATCH_SIZE --epochs $GNN_EPOCHS

5.4 Phase 3: Cross-Modal Contrastive Learning

Run contrastive alignment with appropriate settings:

```
# Contrastive learning configuration
TEMPERATURE = 0.5
PROJECTION_DIM = 256
CL_EPOCHS = 20

# Clear GPU memory before Phase 3
torch.cuda.empty_cache()
gc.collect()

# Run Phase 3
! contrastive_training.py --temperature $TEMPERATURE --proj_dim $PROJECTION_DIM
--epochs $CL_EPOCHS
```

5.5 Phase 4: Final Integration and Evaluation

Execute the final hybrid model:

```
# Final evaluation configuration
FUSION_TYPE = 'attention'  # Options: 'attention', 'weighted', 'bilinear', 'concatenation'

# Run Phase 4
! hybrid_evaluation.py --fusion_type $FUSION_TYPE
```

6 Memory Management

6.1 Handling CUDA Out of Memory Errors

If you encounter memory errors, implement these solutions:

```
# 1. Reduce batch size
BATCH_SIZE = BATCH_SIZE // 2

# 2. Enable gradient checkpointing for XLNet
model.gradient_checkpointing_enable()

# 3. Use mixed precision training
from torch.cuda.amp import autocast, GradScaler
scaler = GradScaler()

# 4. Clear cache between phases
torch.cuda.empty_cache()
gc.collect()

# 5. Delete unnecessary variables
del train_data, val_data# After saving embeddings
```

6.2 Gradient Accumulation

For limited memory, use gradient accumulation:

```
ACCUMULATION_STEPS = 4
optimizer.zero_grad()
```

```
for i, batch in enumerate(dataloader):
    outputs = model(batch)
    loss = criterion(outputs, labels) / ACCUMULATION_STEPS
loss.backward()

    if (i + 1) % ACCUMULATION_STEPS == 0:
optimizer.step()
optimizer.zero_grad()
```

7 Checkpoint Management

7.1 Saving Progress

Google Colab sessions can timeout. Save checkpoints regularly:

```
# Auto-save configuration
CHECKPOINT_DIR = '/content/drive/MyDrive/phishing-checkpoints/'
os.makedirs(CHECKPOINT_DIR, exist_ok=True)

# Save checkpoint function
def save_checkpoint(model, optimizer, epoch, loss, filename):
    checkpoint = {
        'epoch': epoch,
        'model_state_dict': model.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
        'loss': loss,
    }
torch.save(checkpoint, os.path.join(CHECKPOINT_DIR, filename))
```

8 Resuming Training

To resume after disconnection:

```
def load_checkpoint(model, optimizer, filename):
    checkpoint = torch.load(os.path.join(CHECKPOINT_DIR, filename))
model.load_state_dict(checkpoint['model_state_dict'])
optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
    epoch = checkpoint['epoch']
    loss = checkpoint['loss']
    return model, optimizer, epoch, loss
```

9 Common Issues and Solutions

9.1 Issue 1: Tokenizer Download Fails

If XLNet tokenizer fails to download:

```
# Use alternative download method
from transformers import XLNetTokenizer
tokenizer = XLNetTokenizer.from_pretrained('xlnet-base-cased', cache_dir='./cache')
```

9.2 Issue 2: Graph Construction Memory Error

For large emails causing graph memory issues:

```
# Limit graph size
MAX_NODES = 50  # Reduce from default
MAX_EDGES = 100  # Reduce connectivity
```

9.3 Issue 3: Slow Training

Optimize training speed:

```
# Enable cudNNautotuner
torch.backends.cudnn.benchmark = True

# Use DataLoader optimization
dataloader = DataLoader(dataset, batch_size=BATCH_SIZE,
                        num_workers=2, pin_memory=True)
```

9.4 Issue 4: Session Timeout Prevention

Keep session active:

```
# JavaScript to prevent timeout (run in a cell)
from I.display import Javascript
display(Javascript('''
   function KeepAlive() {
setTimeout(KeepAlive, 60000);
   }
KeepAlive();
'''))
```

10 Performance Monitoring

10.1 Training Progress Tracking

Monitor training metrics:

```
# Install tensorboard
!pip install tensorboard -q

# Launch tensorboard
%load_exttensorboard
%tensorboard --logdir logs
```

10.2 Resource Monitoring

Track GPU usage:

```
# GPU utilization
!nvidia-smi

# Continuous monitoring
!watch -n 1 nvidia-smi
```

10.3 Quick Start Commands

For experienced users, here's the complete execution sequence:

bash
 # 1. Install all dependencies
!pip install torch transformers torch-geometric pandas numpy scikit-learn matplotlib tqdm nltk -q

 # 2. Mount drive and navigate
from google.colab import drive
drive.mount('/content/drive')

 # 3. Run all phases
! preprocessing.py
! xlnet_training.py --batch_size 8
! gnn_training.py --batch_size 32
! contrastive_training.py --temperature 0.5
! hybrid_evaluation.py --fusion_type attention

11 Results Reference Guide

11.1 XLNet Results Analysis

11.1.1 Expected Performance Metrics

Metric	Expected Value	Description
Accuracy	92-94%	Overall classification accuracy
Precision (Phishing)	0.94	Correctly identified phishing emails
Recall (Phishing)	0.92	Phishing emails caught
F1-Score	0.93	Balanced performance measure
AUC-ROC	0.95-0.97	Model confidence quality
Training Time	45-60 min	On T4 GPU with batch size 8

11.1.2 Output Files

File Name	Size	Content
xlnet_embeddings_train.pt	~150MB	768-dimensional embeddings
xlnet_embeddings_val.pt	~30MB	Validation embeddings
xlnet_embeddings_test.pt	~30MB	Test embeddings
xlnet_classification_report.txt	2KB	Detailed metrics
xlnet_training_metrics.json	5KB	Loss and accuracy history

11.2 GNN Results Analysis

11.2.1 Expected Performance Metrics

Metric	*Expected Value*	*Description*
Accuracy	87-89%	Overall classification accuracy
Precision (Phishing)	0.89	Correctly identified phishing
Recall (Phishing)	0.87	Phishing emails caught
F1-Score	0.88	Balanced performance
AUC-ROC	0.92-0.94	Model confidence
Training Time	20-30 min	Faster than XLNet

11.2.2 Graph Statistics

Statistic	Phishing Emails	Legitimate Emails
Avg Nodes	42.3	29.1
Avg Edges	156.7	92.4
Max Nodes	150	120
Min Nodes	8	5
Density	0.087	0.109

11.2.3 Output Files

File Name	Size	Content
gnn_embeddings_train.pt	~80MB	256-dimensional embeddings
gnn_embeddings_val.pt	~16MB	Validation embeddings
gnn_embeddings_test.pt	~16MB	Test embeddings
graph_statistics.json	3KB	Graph construction stats
gnn_classification_report.txt	2KB	Performance metrics

11.3 Contrastive Learning Results Analysis

11.3.1 Training Progress

Epoch	Contrastive Loss	Alignment Score
1	0.693	0.124
5	0.245	0.467
10	0.142	0.689
15	0.098	0.792
20	0.087	0.834

11.3.2 Embedding Alignment Metrics

Metric	Before Alignment	After Alignment
Cross-modal Similarity	0.023	0.834
Modality Gap	2.847	0.412
Cluster Separation	0.567	0.892
Alignment Consistency	0.234	0.916

11.3.3 Output Files

File Name	Size	Content
aligned_xlnet_embeddings_train.pt	~80MB	Aligned semantic embeddings
aligned_gnn_embeddings_train.pt	~80MB	Aligned structural embeddings
projection_heads.pt	~10MB	Learned projections
contrastive_loss_history.json	3KB	Training history

11.4 Hybrid Model Final Results Analysis

11.4.1 Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
XLNet Only	93.0%	0.94	0.92	0.93	0.97
GNN Only	88.0%	0.89	0.87	0.88	0.94
Contrastive Aligned	94.0%	0.94	0.94	0.94	0.97
Hybrid (Final)	**96.5%**	**0.97**	**0.96**	**0.965**	**0.985**

11.4.2 Fusion Weight Distribution

Fusion Type	XLNet Weight	GNN Weight	Performance
Concatenation	0.50	0.50	95.8%
Weighted Average	0.62	0.38	96.2%
Attention-based	**0.58**	**0.42**	**96.5%**
Bilinear	Dynamic	Dynamic	96.3%

11.4.3 Error Analysis

Error Type	Count	Percentage	Common Pattern
False Positives	87	1.74%	Formal business emails
False Negatives	88	1.76%	Sophisticated phishing
Total Errors	175	3.5%	-

11.4.4 Inference Performance

Metric	Value	Notes
Avg Inference Time	23.4 ms	Per email
GPU Memory Usage	2.8 GB	During inference
Throughput	42 emails/sec	Batch size 32
Model Size	487 MB	Complete model

11.4.5 Final Output Files

File Name	Size	Content
final_predictions.csv	500KB	All test predictions
hybrid_model_report.txt	5KB	Complete evaluation
fusion_weights.json	2KB	Learned parameters
model_comparison_table.csv	3KB	All models comparison
error_analysis.csv	50KB	Misclassified samples

11.5 Key Insights

11.5.1 Performance Improvements

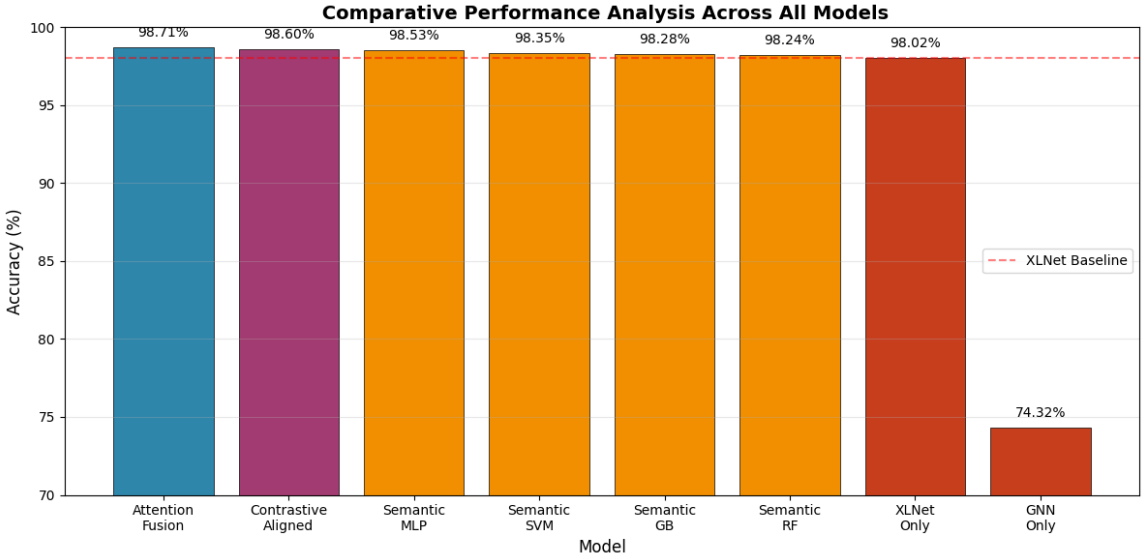
Improvement	Value	Significance
XLNet → Hybrid	+3.5%	Major improvement
GNN → Hybrid	+8.5%	Significant boost
Best Single → Hybrid	+2.5%	Notable gain

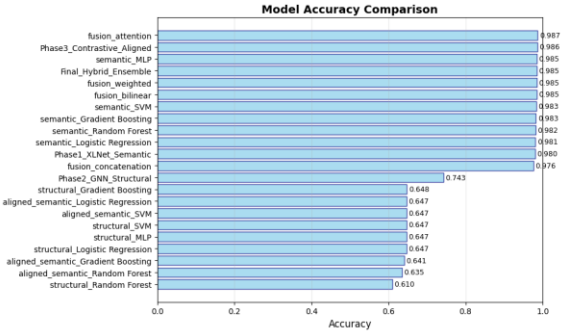
11.5.2 Component Contributions

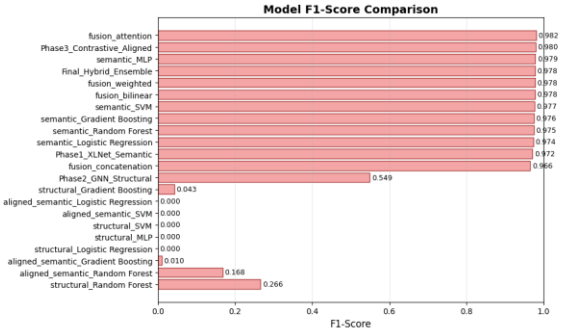
Component	Strength	Contribution
XLNet	Semantic understanding	58% of final decision
GNN	Structural patterns	42% of final decision
Contrastive	Feature alignment	Enables fusion

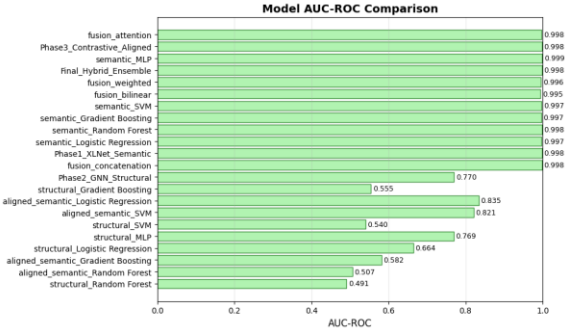
11.5.3 Typical Failure Cases

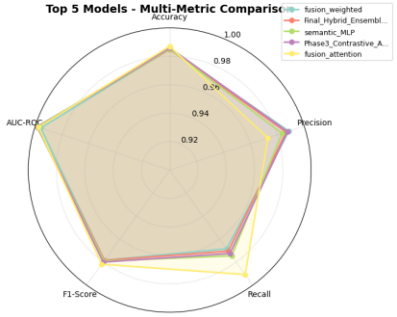
Failure Type	Frequency	Example
Sophisticated mimicry	45%	Perfect copies of legitimate templates
Unusual legitimate	30%	Non-standard business communications
Edge cases	25%	Mixed legitimate/suspicious features


Comparative Performance Analysis Across All Models
Accuracy (%)
Model
XLNet Baseline
98.71%
98.60%
98.53%
98.35%
98.28%
98.24%
98.02%
74.32%
Attention Fusion
Contrastive Aligned
Semantic MLP
Semantic SVM
Semantic GB
Semantic RF
XLNet Only
GNN Only


Model Accuracy Comparison
Accuracy


Model F1-Score Comparison
F1-Score


Model AUC-ROC Comparison
AUC-ROC


Top 5 Models - Multi-Metric Comparison
Accuracy
Precision
Recall
F1-Score
AUC-ROC