National College of Ireland

Configuration Manual

MSc Research Project
MSc in Artificial Intelligence

Jesus Moises Hernandez Lopez
Student ID: 23435526

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Jesus Moises Hernandez Lopez
Student ID:	23435526
Programme:	MSc in Artificial Intelligence
Year:	2025
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	663
Page Count:	7

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:	Jesus Moises
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

Jesus Moises Hernandez Lopez
23435526

1 Introduction

This configuration manual describes the setup, installation, system requirements, and execution for continual learning systems comparing traditional and parameter efficient finetuning methods Yang et al. (2025):

NLP Continual Learning System
(Transformer based text classification with Adapters Houlsby et al. (2019), Prompts, EWC, Replay, DER++ Buzzega et al. (2020), KD)

Vision Continual Learning System
(Vision Transformer with Prompts, Adapters, Replay, and Explainable AI Selvaraju et al. (2017))

These systems implement incremental continual learning and evaluate forgetting, parameter count, runtime, memory consumption, and training stability for NLP and vision tasks. The experiments are executed through configuration files and using automatic checkpointing.

The manual documents:

- Hardware and software requirements
- Installation and environment setup
- Configuration file structure
- Execution commands
- Component descriptions
- Output files

2 System requirements

2.1 Hardware Requirements

The recommended hardware configuration is summarized in Table 1. Note that storage needs may vary depending on how many experiments and checkpoints are retained.

Component	Specification
GPU	**NVIDIA L4 (24 GB VRAM)** – Google Colab Premium
RAM	12–16 GB
Disk	10–20 GB free for project files and dependencies. Additional storage (10–200+ GB) may be required depending on how many experiments, checkpoints, and results are kept. These files can be deleted between runs to reduce disk usage.

Table 1: Recommended System Requirements

2.2 Software Requirements

The project requires the following software packages and libraries:

- Python 3.10 (tested with 3.10+)
- PyTorch 2.9.1
- Transformers 4.57.3
- HuggingFace Datasets 4.4.1
- Accelerate 1.12.0
- PEFT 0.18.0
- Adapter-Transformers / Adapters 1.2.0
- NumPy 2.3.5, Matplotlib 3.10.8, scikit-learn 1.8.0
- OpenCV (opencv-python) 4.12.0.88
- tqdm 4.67.1
- PyYAML 6.0.3

The required Python dependencies can be installed using the following command:

```
pip install torch torchvision transformers datasets accelerate peft \
adapter-transformers pyyaml scikit-learn matplotlib pillow opencv-python tqdm
```

3 Installation and Environment Setup

3.1 Upload the Project to Google Drive

To run the project in Google Colab, upload the project folder to your own Google Drive:

1. Download the project folder (ZIP) from the provided source.
2. Upload it to: `MyDrive/peft_cl`

3. In Google Colab, mount your Drive and navigate to the folder:

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

%cd /content/drive/MyDrive/peft_cl
```

3.2 Installing dependencies

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

or install packages manually.

4 Running the NLP Continual Learning System

This system implements continual learning for text classification using PEFT strategies (Adapters Houlsby et al. (2019), Prompts), Replay and DER++ Buzzega et al. (2020), EWC, and Knowledge Distillation.

4.1 Execution Command

In Colab:

```
!python -m src.train --config configs/sampled_5task.yaml --strategy adapters_replay
```

To resume training from a previously saved checkpoint, specify the strategy and task order:

```
!python -m src.train --config configs/sampled_5task.yaml \
    --resume_from checkpoints/<strategy>/<order>
```

4.2 NLP Configuration File

Below is an example of a configuration file used to define the model backbone, strategy, training settings, replay parameters, adapter configuration, and task order:

```
backbone: "bert-base-uncased"
strategy: "adapters_replay"
data_root: "data_root"

epochs_per_task: 3
batch_size: 16
max_length: 256

replay:
  enabled: true
  buffer_percent: 0.01
  ratio_in_batch: 0.25
```

```
kd:
  enabled: false
  temperature: 2.0
  loss_weight: 0.5

adapters:
  bottleneck: 32
  type: "pfeiffer"

task_orders:
  - ["agnews", "yahoo", "amazon", "dbpedia", "yelp"]

seed: 42
```

4.3 Available Strategies

The following training strategies are supported in the framework for NLP text tasks and are summarized in Table 2.

Strategy	**Description**
ewc	Elastic Weight Consolidation for mitigating catastrophic forgetting in NLP text models.
adapters_replay	Adapter-based training combined with experience replay for NLP text tasks.
prompts_replay	Prompt tuning combined with experience replay for NLP text tasks.
adapters_replay_kd	Adapter-based replay with Knowledge Distillation for NLP text models.
prompts_replay_kd	Prompt-based replay with Knowledge Distillation for NLP text models.
derpp	Replay with stored logits (Dark Experience Replay++) for NLP text continual learning.

Table 2: Overview of Supported Continual Learning Strategies for NLP Text Tasks

5 Running the Vision Continual Learning System

5.1 Execution Command

Example of a vision continual learning experiment using adapters and replay:

```
!python -m src.vision.train_vision --config configs/vision_adapters_replay.yaml
```

5.2 Vision Configuration File

Below is an example configuration file for running vision continual learning experiments using adapters and replay:

```
seed: 42
device: cuda

backbone: "google/vit-base-patch16-224"
strategy: "adapters"

batch_size: 64
epochs: 4
lr: 3.0e-4
weight_decay: 0.01

resume: true
ckpt_dir: "checkpoints_vision_adapters_replay"

replay:
  enabled: true
  max_samples: 2000
  samples_per_class: 20
  ratio: 0.25

adapters:
  bottleneck: 64

vpt:
  length: 20

xai:
  enabled: true
  samples_per_task: 12
  save_dir: "xai_vision_outputs_adapters_replay"
  animate: true
  gif_fps: 1.5
```

5.3 Available Vision Strategies

The following training strategies are supported for vision continual learning and are summarized in Table 3.

Strategy	Description
finetune	Full ViT training on all model parameters.
adapters	Train only adapter modules and the classification head.
vpt	Train Visual Prompt Tuning tokens inserted after the CLS token.
adapters_replay	Adapter-based training combined with experience replay.
prompts_replay	Prompt-tuning combined with experience replay.

Table 3: Vision Continual Learning Strategies

6 Outputs (NLP + Vision)

6.1 Vision Outputs

Running the vision continual learning experiments generates the following outputs:

- `metrics.csv` — Overall metrics per task (accuracy, loss, timing).
- `accuracy_matrix.csv` — Task-by-task accuracy matrix.
- `confusion_taskX.csv` — Confusion matrix for each task.
- Task comparison images — Side-by-side visual comparisons across tasks.
- `gradcam_evolution.gif` — Animated GIF showing explanation evolution across tasks.
- Checkpoints per task — Saved model states for each task and strategy.

6.2 NLP Outputs

Running the NLP continual learning experiments produces the following outputs:

- `results.csv` — Contains accuracy, loss, and per-task performance metrics.
- Forgetting scores — Quantifies performance degradation on earlier tasks over time.
- Checkpoints — Saved model states for each strategy and task order.
- Replay buffer snapshots — Stored samples used during replay-based training.
- Logs — Training logs including loss curves, timing information, memory usage, training time, trainable parameters, and evaluation summaries.

References

Buzzega, P., Boschini, M., Porrello, A., Abati, D. and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline, *Advances in neural information processing systems* **33**: 15920–15930.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M. and Gelly, S. (2019). Parameter-efficient transfer learning for nlp, *International conference on machine learning*, PMLR, pp. 2790–2799.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization, *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.

Yang, Y., Zhou, J., Ding, X., Huai, T., Liu, S., Chen, Q., Xie, Y. and He, L. (2025). Recent advances of foundation language models-based continual learning: A survey, *ACM Computing Surveys* **57**.