National College of Ireland

# Comparative Analysis of Parameter-Efficient Fine-Tuning and Traditional Continual Learning Methods

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:** | Comparative Analysis of Parameter-Efficient Fine-Tuning and Traditional Continual Learning Methods |
| **Word Count:** | 6626 |
| **Page Count:** | 22 |

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): | |

# Comparative Analysis of Parameter-Efficient Fine-Tuning and Traditional Continual Learning Methods

Jesus Moises Hernandez Lopez
23435526

**Abstract**

Continual learning allows models to adapt to new data without being fully retrained. This often produce catastrophic forgetting, where performance on earlier tasks drops when new ones are learned. This thesis studies this problem in class-incremental text classification and a complementary vision experiment. Basically, it compares traditional continual learning baselines with parameter-efficient fine-tuning (PEFT) methods such as adapters and prompt tuning combined with and without replay or knowledge distillation.

The experiments indicate that replay-based PEFT methods retain earlier tasks more effectively and show more stable behaviour over task sequences than regularisation-based approaches and full fine-tuning, while requiring significantly fewer trainable parameters. A complementary study in a vision setting with a Vision Transformer (ViT) shows similar trends, suggesting that these advantages are not limited to text classification. Overall, the results highlight that focusing only on average accuracy can mask substantial degradation on early tasks, and that retention and computational cost should be considered jointly when evaluating continual learning methods.

## 1 Introduction

Continual learning focus on a simple requirement: models should be able to update their knowledge when new data arrives, and it should occur without starting from scratch each time. Usually, a standard supervised model is trained once on a fixed dataset. On the other hand, CL is prepared to obtain a stream of tasks or distribution data over time. In this case, the model learns the new information while performing well on previously learned tasks. This feature is very important when data is constantly updated or when full retraining is considered expensiveYang et al. (2025).

These scenarios are common in natural language processing (NLP) and computer vision AI systems where new information such as topics, domains or categories appears over time. One of the main problems is the catastrophic forgetting. This occurs when the model learns a new task and accuracy on previous tasks drops. This thesis focuses on class incremental learning which occurs when the set of class labels grows while new tasks arrive, and at test time, the model has to predict the correct new class without knowing the task identity. In this setting, the model has to predict both previously seen data and new classes.

The main goal of this thesis is to research how to implement parameter-efficient fine-tuning continual learning strategies and control catastrophic forgetting. CL strategies for neural networks cover replay buffers and regularization methods (for example, Elastic Weight Consolidation (EWC)). Also, it includes distillation based methods such as Learning without Forgetting (LwF) and DER++. These approaches can mitigate forgetting but they need to update the full model parameters, which produces a high use of memory, long training times and bigger development costs.

Parameter efficient fine tuning (PEFT) is an alternative for classic CL methods. These methods such as adapters and prompts tuning, are able to keep the backbone partially or fully frozen and can introduce new parameters specially updated for each new task. On this way, the number of trainable parameters weights is reduced and the training becomes more efficient in terms of time and memory. PEFT techniques were initially used for NLP tasks, after that, they were adapted to vision models. The behavior of these PEFT techniques in combination with replay and distillation is not yet well characterized. In fact, studies made on this matter emphasize accuracy and forgetting, but giving less importance to metrics related to efficiency or performance evolving through different tasks. The main part of this thesis analyzes continual learning techniques for text classification. To evaluate these experiments, five benchmarks datasets (AG News, Yelp Review Full, Amazon Polarity, Yahoo Answers Topics, and DBpedia-14) are used in a class incremental learning. Transformer backbones like BERT, DistilBERT, and RoBERTa are trained with adapters and prompt tuning. Also, replay buffers and knowledge distillation are combined with these techniques to control the forgetting. The experiments have been executed under class incremental condition. So, task identifiers are unknown at test time and each task adds new classes.

Besides NLP study, a complementary vision experiments are considered using Vision Transformer (ViT). These experiments are trained on CIFAR-100 in a class incremental configuration. So, adapters, adapters with replay, prompt tuning (VPT), and prompts with replay are compared to full fine tuning. Additionally, explainable AI is applied using Grad-CAM to visualize which parts of the image drive the model decisions and observe how it shifts while a new task is learned. Also, these experiments help to know whether the patterns in text classification appear in image classification in terms of measuring forgetting when model attention changes over time.

## 1.1 Research Question

> How do parameter-efficient tuning methods (adapters and prompt tuning), when combined with replay and knowledge distillation, compare to traditional continual learning strategies (EWC, LwF, DER++) in terms of catastrophic forgetting and training efficiency (parameter count, runtime, memory consumption, and training stability) in class-incremental text classification tasks?

## 1.2 Contributions

This work makes the following contributions:

- A systematic evaluation of parameter efficient tuning methods (adapters and prompt tuning) for continual learning in text classification. This uses class incremental setup with five NLP datasets.

- A comparison of PEFT methods combined with replay and knowledge distillation to established continual learning strategies (EWC, LwF, DER++). This comparison is based on accuracy, forgetting, backward transfer, and efficiency measures (parameter count, runtime, memory usage, and training stability).
- A complementary vision study on CIFAR-100 with a Vision Transformer. Here, the full fine-tuning, adapters, and prompt tuning (with and without replay) are contrasted to see whether the main trends from the NLP experiments also appear in image classification.
- An experimental framework for continual learning across text and vision that integrates Grad-CAM-based visual explanations for the vision models and links forgetting and stability to changes in model focus.

## 1.3 Report Structure Overview

The remainder of this documents is organized as follow. The introduction chapter presents motivation, problem and objectives. The related work chapter presents classic continual learning methods and modern parameter efficient CL Methods. The methodology chapter presents the design of experiments, datasets, models and evaluation metrics. The design chapter explain how CL strategies and PEFT are implemented in a common framework. The implementation chapter covers all the technical part of the framework. The evaluation chapter analyze and present the results for text and vision experiments. Finally, the conclusion chapter present the main findings and future work.

# 2 Related Work

This section reviews the progress of continual learning methods from traditional baselines (replay, regularization and distillation) to new parameter efficient fine tuning (PEFT) approaches. The literature review is divided in two major families: classical baselines and modern PEFT based methods.

## 2.1 Classical Continual Learning Baselines

### 2.1.1 Replay and Distillation

Replay and distillation techniques are strong baselines. iCaRL (Rebuffi et al.; 2017) presented replay using a nearest class mean classifier. LwF (Li and Hoiem; 2018) used knowledge distillation but it is not saving samples. It keeps 95% of the previous task accuracy and improves the new tasks by +4–6%. GEM (Lopez-Paz and Ranzato; 2017) applied restrictions on gradient updates to keep the knowledge but it has a higher computational cost. In general, these methods show a good balance between memory use and robustness but they are not parameter efficient. Simple experience replay (ER) also has shown important results and it has been considered as an strong baselines. Chaudhry et al. (2019).

### 2.1.2 Regularization-Based Methods

EWC (Kirkpatrick et al.; 2017)restricts updates in the most important parameters using Fisher information matrix. This helps to reduce forgetting by 15% on Split-MNIST. In general, these approaches help avoid saving a lot of samples. However, they don't scale well to large models.

### 2.1.3 Modern Replay Baselines

Recent replay techniquesimproved the stability and scalability. DER++ (Buzzega et al.; 2020) combined two techniques, replay and distillation. This improved accuracy by +3–6% more than ER which reduces forgetting by 20%. SER (Zhuo et al.; 2023) used forward/backward consistency and X-DER (Boschini et al.; 2023) improved the efficiency and also the performance. In general, even with strong accuracy, these methods still use memory buffers which create privacy and storage issues.

## 2.2 Modern Parameter-Efficient CL Methods

### 2.2.1 Prompt-Based Continual Learning

Prompt tuning reduces the parameters to be trained by freezing the backbone. L2P (Wang et al.; 2021) presented a prompt pool that selects prompts per input. DualPrompt (Wang et al.; 2022) separated task specific and task variant prompts. It improved accuracy by +5.1% more than L2P with 0.6% extra parameters. Prompt tuning does not use memory buffers and provides strong efficient. However, the performance can drop when tasks are complex. These methods are mostly evaluated on vision benchmarks such as class incremental variants of ImageNet.

### 2.2.2 Adapter-Based PEFT

Adapters (Houlsby et al.; 2019) insert lightweight trainable modules in the transformers layers. It achieved similar full fine tuning accuracy but only training 3–5% of parameters. Compacter (Mahabadi et al.; 2021) reduced redundancy via hypernetworks and HyperFormer++. This improved cross task transfer by 2–3% with only 0.29% trainable parameters per task. So, adapters allow scalable CL with no need of storing samples unlike replay baselines, but replay can be better at performance when there is enough memory.

## 2.3 Continual Learning in NLP

NLP CL methods mainly focus on representation robustness. RepCL (Song et al.; 2023) shows contrastive and generative objectives. It improved accuracy by +2–4% and reduced forgetting by 20%. There are other approaches that disentangle generic/specific features or they apply gating for selective activation. So, first these models try to improve their representations rather than use replay or regularization. Other recent works experiment with continual learning using transformer based language models for classification and sequence labelling. However, most of the approaches prefer full fine tuning of the backbone and few times they report efficiency metrics such as parameter count or memory use.

## 2.4 Interpretability in Continual Learning

Interpretability tools like Grad-CAM Grad-CAM (Selvaraju et al.; 2017), attention rollout, and Integrated Gradients (Qi et al.; 2020) are useful to know which information is forgetten. So, using explainable AI with PEFT allows to analyze how prompts or adapters can affect knowledge retention.

## 2.5 Summary of Continual Learning Methods

Table 1 shows important continual learning approaches mainly focused on their methods, metrics, baselines and findings. This is organized showing the evolution from replay and regularization techniques to more recent such as adapters and prompt based strategies.

Table 1: Key continual learning methods included in this thesis.

| **Paper (Year)** | **Method** | **Metrics** | **Baselines** | **Findings** |
|---|---|---|---|---|
| **EWC (2017)** | Regularization; no replay | Average accuracy; forgetting | SGD; multitask | Reduces forgetting by about 15% on Split-MNIST and does not require storing past training examples. |
| **LwF (2018)** | Knowledge distillation; no replay | Old/new task accuracy | Fine-tuning; EWC | Keeps around 95% of the accuracy on old tasks and improves new-task performance by 4–6 percentage points compared with simple fine-tuning. |
| **DER++ (2020)** | Replay + logit consistency | Average accuracy; forgetting | ER; A-GEM; LwF | Improves accuracy by 3–6 percentage points over ER and reduces forgetting by roughly 20%. |
| **L2P (2022)** | Prompt-based (no replay) | Average accuracy; forgetting | DER++; ER; iCaRL | Outperforms DER++ by 4–6 percentage points in average accuracy and lowers forgetting by about 3% on Split ImageNet-R. |

*Continued on next page*

*Table 1 (continued)*

| Paper (Year) | Method | Metrics | Baselines | Findings |
|---|---|---|---|---|
| **DualPrompt (2022)** | Task-general + task-specific prompts | Accuracy; backward transfer | L2P; DER++ | Provides a 5.1-point accuracy gain over L2P, reduces forgetting by 2.4 points, and adds less than 0.6% additional parameters. |
| **Houlsby Adapters (2019)** | Adapter-based PEFT | Task accuracy; parameter efficiency | Full fine-tuning | Stays within 0.4 percentage points of full fine-tuning while updating only 3.6% of the model parameters. |
| **HyperFormer++ (2021)** | Adapters + hypernetworks | Accuracy; transfer; PEFT | Houlsby; fine-tuning | Improves transfer performance by 2–3 percentage points and uses only 0.29% trainable parameters per task. |
| **RepCL (2023)** | Contrastive + generative replay | Average accuracy; forgetting | CRECL; IDBR | Increases accuracy by 2–4 percentage points and reduces forgetting by about 20% on MAVEN and TACRED. |

## 2.6 Research Niche

Traditional baselines (iCaRL, LwF, EWC) are effective but expensive in terms of computational resources. Replay-based baselines (DER++, SER, X-DER) got strong accuracy but they require memory buffers. PEFT methods (prompts, adapters) improved efficiency and removed the privacy issues. In general, adapters show strong stability in vision tasks and prompt methods allows high modularity.

Recent work started analyzing the adapter-based continual learning with pre-trained transformers for both text and image classification focusing on memory efficiency Ermis et al. (2022). This work specifically focusses on memory efficiency. However, the integration of PEFT with replay/distillation is still underexplored. This thesis investigates hybrid PEFT settings and compares them to traditional baselines in NLP and vision experiments under CIL.

# 3 Methodology

This section shows the methodology used to study continual learning with parameter efficient methods in text and vision. All experiments are executed in a class incremental scenario. Each step $t$ the model is trained on the current task but it is evaluated on all tasks seen so far. This allows to measure performance and forgetting from earlier tasks.

## 3.1 NLP Experiments

### 3.1.1 Tasks and datasets

Five benchmarks' datasets are used for text classification. In here, each dataset represents one task AG News (news topic classification), Yelp Review Full (sentiment rating), Amazon Polarity (binary sentiment), Yahoo Answers Topics (question topic classification), and DBpedia (ontology-based entity classification) (Zhang et al.; 2015). The experiments test 3 task orders: AG News → Yelp Review Full → Amazon Polarity → Yahoo Answers Topics → DBpedia; Yelp Review Full → Yahoo Answers Topics → Amazon Polarity → DBpedia → AG News; and DBpedia → Yahoo Answers Topics → AG News → Amazon Polarity → Yelp Review Full. On this way, results are based on the 3 permutations average to avoid unfair comparisons because of task ordering.

### 3.1.2 Backbone and parameter-efficient tuning

For these experiments, 3 pretrained transformer models, bert-base-uncased (Devlin et al.; 2019), distilbert-base-uncased (Sanh et al.; 2019), and roberta-base (Liu et al.; 2019). The inputs are tokenized with the matching tokenizer and truncate them to 256 tokens. Two families of PEFT methods were used. First, Pfeiffer-style adapters to each transformer layer with bottleneck dimensions set per backbone. Only adapter parameters and the classification head are trained. Second, soft prompts of length 20 are added prepended to the input representation. In this setting, prompt embeddings and the classification head are trainable and they keep the backbone frozen. Experiments combined these approaches (adapters and prompts) with replay and knowledge distillation (optional). After that, they are compared to baselines such as under similar training settings for all backbones.

### 3.1.3 Continual learning protocol

Experience replay is applied using a shared configuration where once the memory buffer is enabled, it saves up to 1% of the full training data (buffer percent = 0.01). When training occurs on task $t$, mini batches mix the current task examples with buffer examples. Replay ratio used in this process is 0.25 per batch and random selection from the buffer (selection = random). Using only experience replay with a small episodic memory has been shown strong results and it is considered as strong baseline Chaudhry et al. (2019). Knowledge distillation is an optional feature that can be applied with temperature 2.0 and loss weight 0.5.

### 3.1.4 Training and evaluation

Each model is trained with mini batch gradient decent on an NVIDIA L4 GPU (24 GB VRAM) in Google Colab, which is exposed as a `cuda` device in the configuration. AdamW is only used for the trainable parameters. The weight decay set is 0.001. The

linear learning rates are schedule without warmup. Batch sizes used range between 16 to 48 depending on the backbone. Availability GPU memory also is considered (e.g., 32 for BERT and RoBERTa, 48 for DistilBERT). The training runs 4 epochs per task and learning rates are applied to the classification head and parameter efficient components. `lr_head` is fixed at $5 \cdot 10^{-4}$, while `lr_peft` lies in the range $2\text{–}3 \cdot 10^{-4}$ depending on the backbones to avoid differences in model size and stability. All runs use a fixed random seed (42). The data from datasets is loaded from a common root directory (data/sampled) and the splits are fixed. Train and validation sets are balanced for each dataset by sampling 2000 examples per class and downsample the test set to about 7600 in total. Also, stratified by class.

After each task, the accuracy is calculated on all tasks seen so far. The results are logged with metrics such as training time, evaluation time, throughput (steps per second), peak GPU memory, total parameters, and trainable parameters. Accuracy sequences, final average accuracy, task wise forgetting, and backward transfer are calculated and presented in the evaluation chapter (Section 6).

## 3.2 Vision Experiments

### 3.2.1 Tasks and dataset

The vision setting uses CIFAR-100 (Krizhevsky et al.; 2009) in class incremental learning configuration. The 100 classes are divided between the 5 tasks (20 classes each). So, during each step $t$ the model can see the images from the current 20 classes subset. After that, it is evaluated on all tasks seen so far. In this setting, the task order is fixed for all methods.

### 3.2.2 Backbone and parameter-efficient tuning

The backbone is a Vision Transformer `google/vit-base-patch16-224`. This is initialized from a pretrained checkpoint. The images are resized to $224 \times 224$ pixels. Also, they are normalized with standard ImageNet static and a crop and horizontal flip is applied randomly for each image. Many strategies are considered in this setting. Finetuning strategy updates all backbone parameters and the classification head. Adapters setting, the bottleneck adapters with dimension 64 in each block transformer but the adapters and classification are the only trained. There is a variant combining adapters and experience replay. For prompt tuning (VPT), soft visual prompt length 20 are added to the ViT input sequence. After that, prompts and the head are trained together. In the meantime, backbone stays frozen Also, there is a final variant combining VPT and replay.

### 3.2.3 Continual learning with replay

In the replay setting, the memory has a fixed size. The configuration used for vision experiments has a maximum of 2000 stored examples which is up to 20 examples per class. The overall ratio is 0.25 relative to the current task data. In the training, batches have a mix between current task examples and the replay ones. On this way, the model can see older classes again.

### 3.2.4 Training and evaluation

Visions models are trained with batch size 64, 4 epochs per task and using AdamW. The learning rate is $3 \cdot 10^{-4}$ and weight decay 0.01. All the experiments executed in a NVIDIA L4 GPU (24 GB VRAM) in Google Colab (reported as a `cuda` device in the configuration). The established seed is 42. As in the NLP experiments, the accuracy is calculated on all tasks and after each task. Average final accuracy, forgetting, and backward transfer are obtained from these values. In addition, confusion matrix and Grad-CAM visualizations are created to observe how the focus change or evolve through image regions over the sequence.

# 4 Design Specification

The system is designed considering flexibility and modular as main features. This is useful to evaluate both NLP and vision continual learning methods. So, each strategy either traditional method or parameter efficient approach such as adapters and prompts are executed using a pipeline which does not need changes on the training code to execute each method. On this way, all the methods can be compared applying the same conditions and observe differences between the continual learning strategy and not for the implementation. The overview of this framework is illustrated in Figure 1.

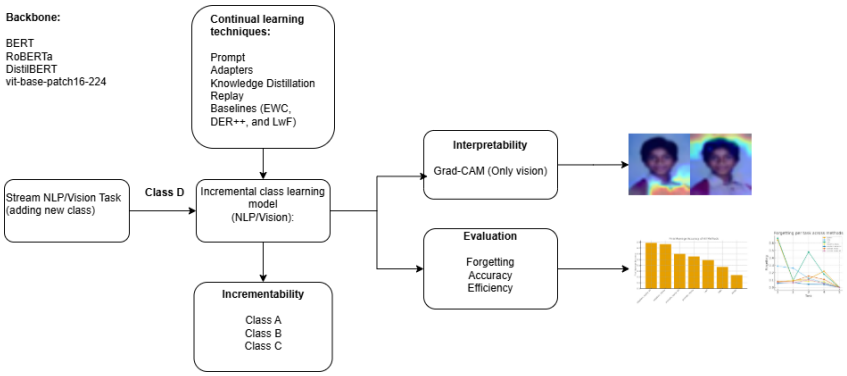



Figure 1: Overview of the proposed modular continual learning framework for NLP and vision.

Backbones models are considered as the main component of this design. For NLP experiments the pretrained models (bert-base-uncased, distilbert-base-uncased, and roberta-base) described in Section 3.1.2 are loaded using HuggingFace library. Also, they are configured to manage the maximum number of classes. So, another important feature is the framework supports fully train the backbones for methods like finetuning, replay and regularization baseline methods methods (DER++, LwF, EWC). Also, it helps frozen backbones for parameter efficient methods like adapters and prompt tuning. The vision pipeline is designed with the same structure but using ViT backbone ViT-Base Patch-16 introduced in Section 3.2.2.

The continual learning strategies are implemented as modules able to plug into the training loop with a common interface. EWC add fisher based regularization term penalizing changes in parameters considered important from the tasks in the past. LwF adds distillation lost but keeping the compatibility with earlier predictions. Replay based module shows memory buffer saving samples from previous tasks. DER++ can improve

replay based by saving logits alongside raw examples. Parameter efficient methods like adapters insert small bottleneck layers in the transformer architectures. Prompt based methods introduce trainable vectors in the input representation. On this way, each module can be activated, deactivated or combine without modifying the training code.

Data is also consistently handled across all methods. For NLP experiments, HuggingFace tokenisers transform the raw text in inputs ready to be processed by the model. It uses a preprocessing mechanism used for all datasets. In vision tasks, Torchvision load, resize, normalize and batch the images. To allow fisher matrices and replay buffers updating internal structures, there is a task manager which control the sequence of tasks. So, this keep class mappings and signals task boundaries to the continual learning modules.

The training process is able to coordinate different model components. For instance, the training is responsible of many aspects such as epoch iteration, loss computation, integrate replay or regularization when they are activated. Also, gradient updates and checkpoints. Another important task from training is the control of parameter freezing rules to make sure only considered parameters are updated for adapter and prompt base configurations. Once a task finish, the training has to invoke shared evaluation routines, logs accuracy on all seen tasks and record important efficiency metrics such as runtime and memory usage.

The vision design contains some extra features focused on explainable AI using Grad-CAM heatmaps. These are generated after each task to observe how the focus in images change when new classes are introduced. Also, confusion matrixes are used to visualize which classes degrade the most after each training. All these components are implemented in the evaluation pipeline.

Finally, for each run the framework track the total number of parameters, number of trainable parameters, the fraction of trainable parameters, peak GPU memory usage and the training time per task. This allows an easy comparison in terms of accuracy, forgetting behavior, and computational cost for all methods in a well structure logging format.

In general, the design focuses on modularity, fairness, reproducibility and extensibility. On this way, continual learning strategies are implemented as modules which can be replace either in NLP or vision pipelines. So, this allows to have a stable foundation for the experiments and making the addition of new methods or backbones easier in future work.

# 5 Implementation

The implementation of this project is an experimental framework which runs continual learning models on text and vision tasks. This section mainly focused on the final system and how it is used in the experiments. The code has different python packages for data loading, model configuration, continual learning strategies, training and evaluation. NLP and vision pipelines have a different implementation but they follow the same patterns like looping over tasks, strategy configuration files (YAML) and a similar structure for logging and checkpoints.

The experiments are implemented using Python with Pytorch as main languages. Transformer and vision transformer are loaded using HuggingFace. This is useful since it gives the pretrained weights, tokenizers and tools to be able to manage model

configurations. Each run or experiment executed is manage using a YAML configuration file which specify the backbone (BERT, DistilBERT, RoBERTa, or ViT). Also, it contains the continual learning strategy (such as EWC, replay, DER++, adapters, or prompts), the order dataset and optimization settings. In case a different method or backbone needs to be executed, changing the YAML config file is enough. This is useful to avoid touching the training code in the NLP or vision pipeline. An excerpt of configuration options for the vision experiments is shown in Listing 1.

Listing 1: Excerpt of the YAML configuration for a ViT vision experiment with 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
replay:
  enabled: true
  max_samples: 2000
  samples_per_class: 20
  ratio: 0.25
adapters:
  bottleneck: 64
```

A big part of the implementation requires to prepare and transform datasets to use them in the class incremental setting. In the NLP settings, each dataset (AG News, Yelp Review Full, Amazon Polarity, Yahoo Answers Topics, and DBpedia-14) , as described in Section 3.1.1, is processed by applying tokenization and removing unused fields. Also, remapping label is also important since each task uses its own distinct class space. The NLP pipeline handle the dataset splits automatically. This makes sure the data is consistent in for all methods every time they are running. In the vision settings, as described in section 3.2.1, CIFAR 100 is split in 5 subsets. Each subset has 20 classes for each separate task.

The modelling component helps to have together the traditional CL strategies (EWC, LwF-style distillation, replay, and DER++) and parameter efficient tuning methods (adapters and prompts). Since components are implemented in a modular way, it is possible to combine any backbone with any compatible continual learning strategy. So, this helps to broad the exploration and avoid rewriting or make changes in the main training loop. Another important feature of this is the internal parameter counting that calculates the total and trainable parameters counts. On this way, if a new configuration is created, it allows a comparison between model sizes update budges for the different methods.

The training loop contains the continual learning circle. It loads the current task, adjust the classifier head, train on different epochs, apply the necessary losses by the continual learning strategy and also evaluates the model on all tasks seen so far. If Reply is enabled in the configuration file, it is implemented using memory buffers that save samples (in case of DER++, it saves logits) from tasks previously executed and mixing

them with the current batches based on the configure ratio defined in the config file as well. If EWC is activated, it uses fisher estimation after each task to apply the penalty in the next task. Each step is registered in logs. This includes loss values, accuracy progression, runtime and peak memory usage. Also, the implementation generates learning curves.

The system generates different outputs to represent the quantitative analysis. When a task is completed, the checkpoints are saved. These checkpoints contained model weights, adapter or prompt parameters in case they are used. Also, replay buffer state is storage. Through the logging, the system writes CSV files saving data like accuracy per task, final average accuracy, forgetting scores backward transfer, training time, evaluation time, and memory usage. So, logs are the main resource to save data related to performance and efficiency for all methods.

Besides numeric metrics, the implementation also generates visual explanations only for vision experiments. After each task, Grad-CAM heatmaps is used to visualize which image regions influence the model to make predictions. Also, it helps to know how region can change when new tasks added. Confusion matrices are calculated per task to know exactly which classes are more affected by forgetting.

In general, the implementation generates many types of outputs, including trained model checkpoints, replay buffers, CSV logs, metric tables, performance plots and Grad-CAM outputs. All this information is important to make a proper analysis and show the evidence from the conclusions presented in this thesis. The system reproducible using the saved configuration files. Also, the fixed random seeds and checkpoint models allows to replicate the experiments using the same conditions.

# 6 Evaluation

This chapter shows the results for all the experiments in NLP and vision. The comparison between traditional continual learning strategies and parameter efficient methods focus on how good the models are at keeping previous learned knowledge. Also, stability for all tasks and how the performance can change when the number of parameters is reduced (Buzzega et al.; 2020; Boschini et al.; 2023; Houlsby et al.; 2019; Mahabadi et al.; 2021). The analysis shows accuracy progression, forgetting, backward transfer and efficiency metrics. These metrics are runtime, memory usage, and trainable parameters counts. Also, plots tables and visual explanations are added for the analysis.

## 6.1 NLP Evaluation

NLP experiments are the main contribution on this thesis. As it was seen in the research question (Section 1.3), it focuse on continual learning for text classification. The experiments are based on transformers based models. These experiments can be seen in a more detailed context in papers focus on pretrained language models for continual text classification (Song et al.; 2023; Huang et al.; 2021; Ahrens et al.; 2021).

## 6.2 Accuracy Progression

Figure 2 shows accuracy changes over the 5 tasks for each method. As can be seen, Replay-based PEFT approaches like adapters with replay (with and without knowledge distilation) keeps high and stable accuracy. An small drop can be seen in the last tasks. This results are in line with previous findings that show replay base methods are good

baselines for class incremental context (Rebuffi et al.; 2017; Buzzega et al.; 2020; Boschini et al.; 2023). Prompts with replay (with and without knowledge distilation) methods are in second place. They are located between adapters and regularization baselines. So, EWC and DER++ have lower performance and LwF is weaker. It has an small recovery on the final tasks. In general, the curves shows replay base PEFT methods perform better than regularization strategies working alone.

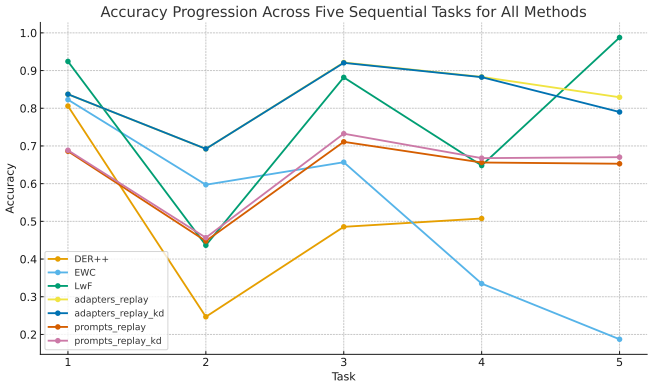



Figure 2: Accuracy progression across five sequential tasks for all evaluated continual learning methods.

## 6.3 Final Average Accuracy

Figure 3 shows the final average accuracy when 5 tasks are executed. Replay based PEFT methods got the strongest results. Particularly, adapters with replay combined with knowledge distillation got the highest average accuracy. After that, the second best is the prompt with replay methods. The regularization based baselines (LwF and EWC) got lower averages (Li and Hoiem; 2018; Kirkpatrick et al.; 2017). DER++ has a bad performance overall amount the methods tested (Buzzega et al.; 2020). So, there is a pattern that continue showing replay is better than regularization alone when is about keeping the performance over time.

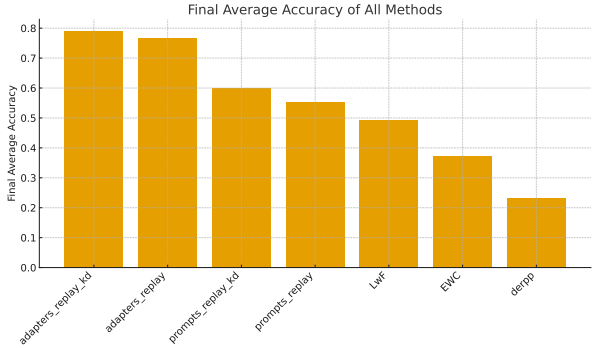
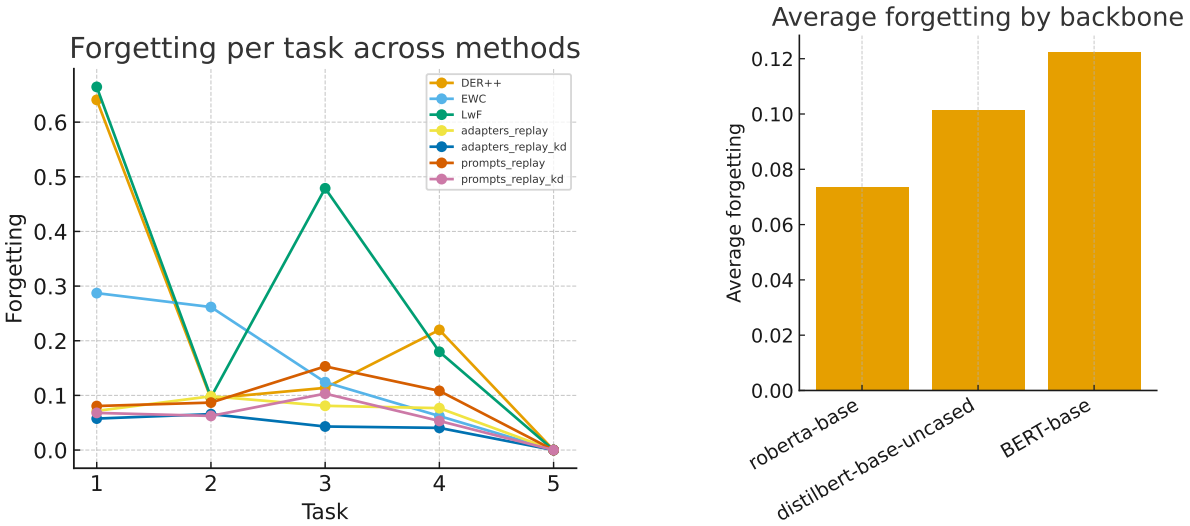



Figure 3: Final average accuracy of all methods after completing all tasks.

## 6.4 Forgetting Behaviour

Figure 4 shows the forgetting for all the methods and backbones. DER++ and LwF have the most unstable forgetting. EWC reduces the forgetting partially only on earlier tasks. Replay based methods like adapters_replay_kd keep the forgetting low. This supports the findings on adapter style parameter efficient tuning (Houlsby et al.; 2019; Mahabadi et al.; 2021). RoBERTA pretrained models are more robust than DistilBERT and BERT base. In general, PEFT methods with replay and stronger backbones are important to keep a good performance over time.




(a) Forgetting per task across methods.

(b) Average forgetting by backbone.

Figure 4: Forgetting behaviour across continual learning methods (left) and backbone models (right).

## 6.5 Backward Transfer

Figure 5 shows the backward transfer (BWT) scores for all the methods. The regularization base strategies such as EWC, DER++ and LwF have negative BWT values. So, this means that every time the model learns new tasks, its performance gets worse on earlier tasks. On the other hand, replay based PEFT methods shows almost a neutral BWT. This indicates that replay and PEFT keep the performance on past tasks. Sometimes, it can even show a small improvement.

## 6.6 Efficiency Analysis

Table 2 shows the computational cost for different methods. There are approaches that update all the parameters (DER++, LwF, EWC). So they use more trainable weights and GPU memory than adapters and prompts variants. Replay based PEFT methods reduce this amount from almost 100% to roughly 1–2%. This reinforce the earlier works on adapters and prompts base tunning training only a small amount of parameters updated (Houlsby et al.; 2019; Mahabadi et al.; 2021; Wang et al.; 2021, 2022). It shows clearly the efficiency for PEFT style approaches.

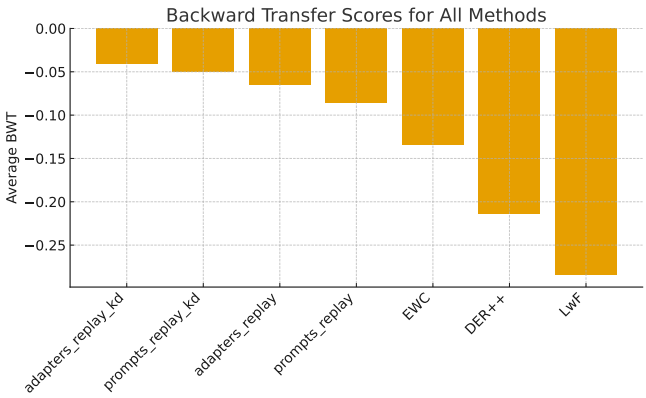



Figure 5: Backward transfer (BWT) scores for each continual learning method.

Table 2: Efficiency metrics for all methods.

| Strategy | Train Time | Eval Time | Max GPU Mem | Total Params | Trainable Params | Trainable % |
|---|---|---|---|---|---|---|
| DER++ | 972.07 | 93.07 | 5557.67 | 6.70e+07 | 6.70e+07 | 1.00e+00 |
| EWC | 3210.90 | 6563.68 | 6844.83 | 3.35e+07 | 3.35e+07 | 9.84e-01 |
| LwF | 11819.20 | 171.90 | 8317.50 | 1.18e+08 | 1.18e+08 | 1.00e+00 |
| adapters_replay | 1841.17 | 304.57 | 3393.67 | 1.02e+08 | 1.59e+06 | 1.64e-02 |
| adapters_replay_kd | 2021.30 | 310.63 | 5846.89 | 1.02e+08 | 1.58e+06 | 1.64e-02 |
| prompts_replay | 1808.07 | 198.20 | 2236.13 | 5.90e+07 | 4.15e+05 | 3.39e-01 |
| prompts_replay_kd | 2027.88 | 300.79 | 5329.22 | 1.01e+08 | 9.42e+05 | 1.04e-02 |

## 6.7 Stability of Accuracy Across Tasks

Figure 6 compares accuracy trajectories. These accuracies are grouped by methods as stable or unstable for all the tasks. So, the replay base methods are the stable group and keep a high accuracy and almost constant when new task are added. On the other hand, regularization methods show a lower accuracy mainly on last tasks. In general, this shows that replay technique has an important role at preventing errors appear over time.

## 6.8 Vision Experiments Evaluation

The vision experiments complement the NLP study. This study analyzes continual learning with vision transformers (ViTs), which regularly are prone to forget in early tasks. The methods considered are full fine tuning, adapters, adapters with replay, prompt tuning (VPT), and prompt with replay. Quantitative results and explainability visualizations are used to observed the methods behavior.

### 6.8.1 Accuracy and Forgetting in Image Tasks

The average accuracy shows that fine tuning has the highest average score ($\approx 0.97$). However, this can be misleading since it does not reflect the accuracy decline on earlier task. So, the progression of the plot in Figure 7b shows finetuning forget the most in

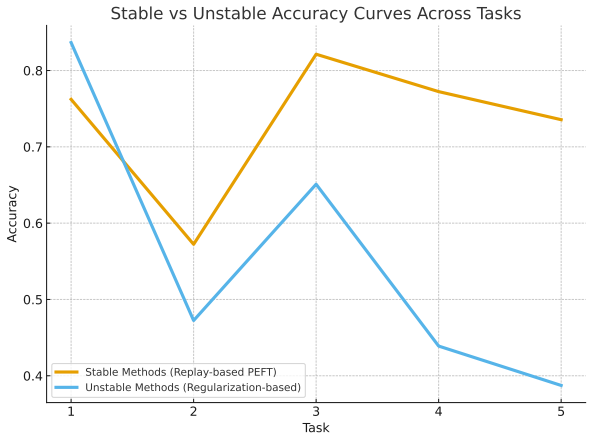
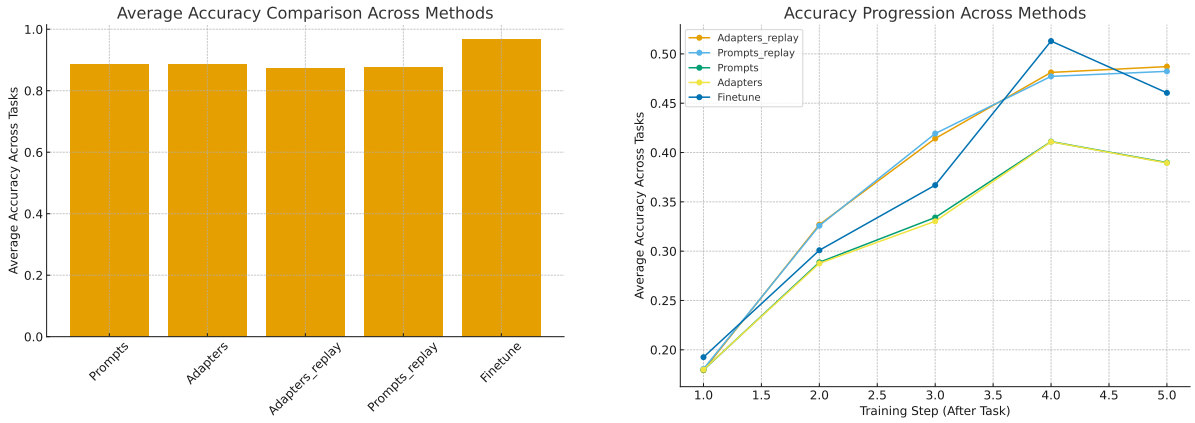



Figure 6: Stable (replay-based PEFT) versus unstable (regularization-based) accuracy curves across tasks.

considering all the sequence task. The replay based methods got average values around 0.88 keeping an stable performance and preserving the earlier knowledge. The methods where replay is not used such as adapter and prompt alone obtained similar averages ($\approx 0.89$) but it can vary more for all tasks. These results show that having the highest accuracy does not mean good retention, and using replay get more stable learning.

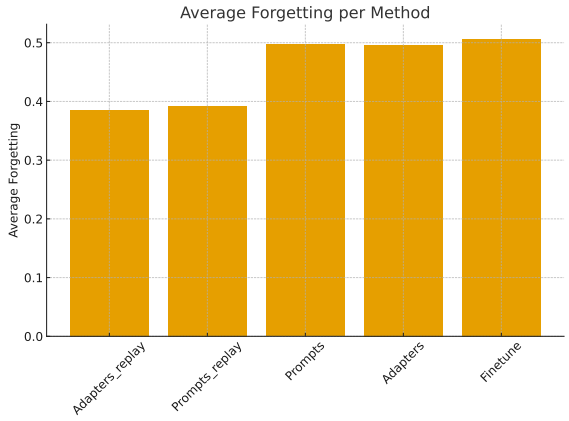



(a) Average accuracy across methods.

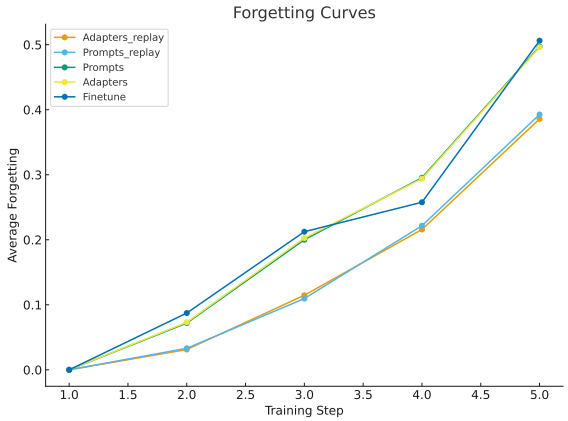
(b) Accuracy progression across tasks.

Figure 7: Comparison of continual learning methods using mean accuracy (left) and accuracy progression (right).

The forgetting results are shown in Figure 8a and Figure 8b show finetuning loses performance in the earlier tasks (average forgetting above 0.50). The prompt and adapters alone methods also have a similar behavior. On the other hand, replay adapters and prompt methods shows that the earlier knowledge is kept effectively. The average forgetting from these methods is around 0.39m and it has visible flat forgetting curves. These results shows that replay is the main factor to reduce forgetting across all task sequence.




(a) Average forgetting across methods.




(b) Forgetting curves across training steps.

Figure 8: Forgetting comparison of continual learning methods using mean forgetting (left) and forgetting progression (right).

Figure 9 shows that all methods tested have negative values for back transfer. So, new knowledge always reduce the performance on earlier tasks. The most negative values are from finetuning and non replay prompt and adapters methods (around $-0.63$ and $-0.62$). The methods using replay are less negative (about $-0.49$ and $-0.48$). This indicates the inference with past tasks is weak.

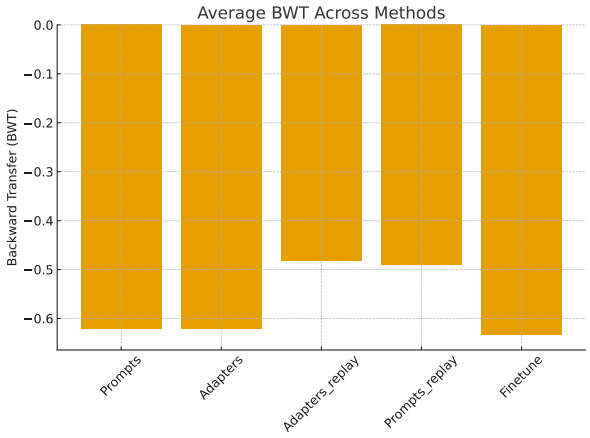
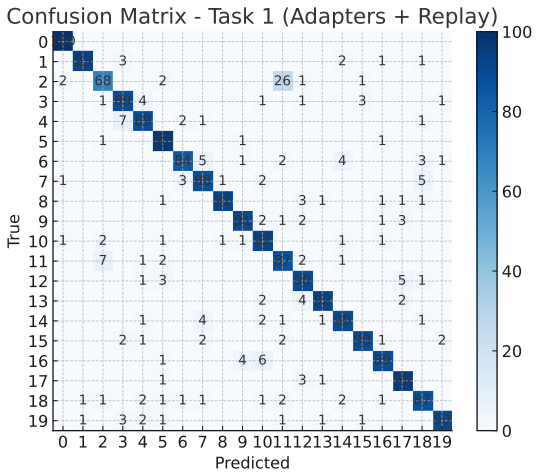


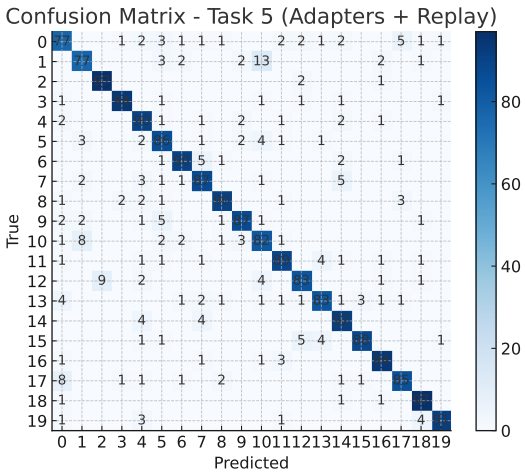

Figure 9: Backward transfer (BWT) scores across methods on the vision tasks.

### 6.8.2 Confusion Matrix Analysis

Figure 10 shows confusion matrix from task 1 and task 5 from Adapters + Replay method. For task 1, the matrix is diagonal. This means classes are predicted properly after training. For task 5 the diagonal is still strong after learning all tasks. This means classes are still predicted well but there is a small increase of entries off diagonal. This matches the small forgetting seen for adapters + replay in the previous results.




(a) Task 1 (Adapters + Replay).




(b) Task 5 (Adapters + Replay).

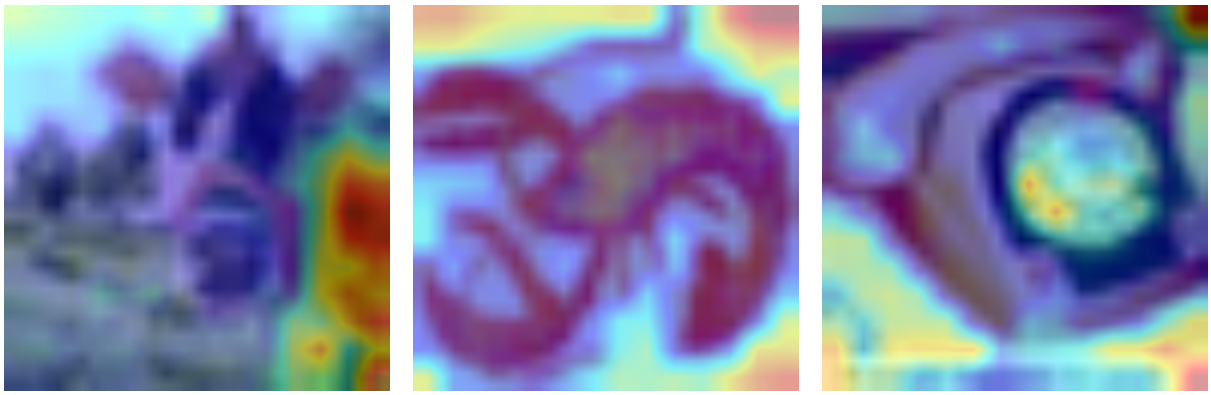
Figure 10: Confusion matrices for Adapters + Replay on Task 1 (left) and Task 5 (right).

## 6.9 Explainability Analysis

This section uses Grad-CAM (Selvaraju et al.; 2017) as explainable AI method. This method shows in a visual way the regions from images that influence the predictions. This is use to know where the model focuses in a task and also observed how it changes after each task.

### 6.9.1 Within Task Explanations

Figure 11 shows 3 random Grad-CAM maps samples for Adapters + Replay. The highlighted regions are not focus on the background but closer to the object. This means the model makes its decisions based on parts of the image.



(a) Task 1. (b) Task 3. (c) Task 5.

Figure 11: Grad-CAM heatmaps for Adapters + Replay on three tasks.

### 6.9.2 Cross-Task Representational Drift

To observe how explanations change, Figure 12 compares Grad CAM maps generated from the task 1 image and the same image after the trained is completed by all tasks. Task 1 is on the left side and task 5 on the right side. The example from Adapters +

Replay is focused all the time in the person across task 1 and task 5. In finetuning, the map generated from task 5 is more diffuse and the attention shifts in a different position away from the original one.

(a) Adapters + Replay: Task 1 (left) vs Task 5 (right).

(b) Finetuning: Task 1 (left) vs Task 5 (right).

Figure 12: Cross-task Grad-CAM comparison for a single image.

# 7 Discussion

Both text and vision experiment similar patterns are observed. The methods that update all parameters like finetuning and regularization base methods shows high average accuracy in the last tasks. However, the accuracy in earlier tasks drops strongly. It is observed that progression and forgetting curves shows that these high accuracies on last tasks can be misleading since in the initial ones, the knowledge is overwritten. On the other hand, methods based on replay and PEFT like adapters and replay (with or without knowledge distillation) can keep a more stable accuracy and limit the losses on the first tasks. Forgetting and backward transfer results support this view. The strategies such as DER++ and LwF shows the highest forgetting and most negative values. EWC reduce the forgetting partially. Replay based PEFT methods presented lower forgetting. Also, they show less negative and sometimes close to a neutral backward transfer value. This indicates replay is more effective than regularization methods alone when keeping the old knowledge is the goal. So, stronger backbones like RoBERTa in NLP and ViT in vision can have a good performance and get the benefits from replay and parameter efficient updates.

Metrics related to efficiency support also this conclusion. Full parameter methods need more memory and computation than adapters and prompt approaches. Replay based PEFT reduce the trainable parameters used to a small percentage of the total and many times overcoming the stability of full finetuning. Vision experiments on CIFAR 100 show the same idea. Finetuning achieves the highest mean accuracy and the worst forgetting and BWT. So, replay based adapters and prompts have a better balance considering keeping knowledge and cost.

In general, these results give an answer to the research question in favor of replay based PEFT methods. So, in the context of class incremental settings, adapters and prompts with replay and with/without knowledge distillation have better results for accuracy, forgetting and resource usage than EWC, LwF , DER++ or finetuning. So, this support the idea that if using large transformer models with resource constraints, PEFT with replay is a strong option for continual learning approaches.

# 8 Conclusions and Future Work

This thesis studied continual learning with parameter efficient methods in NLP and vision. Replay based PEFT methods such as adapters and prompts with replay (with/without knowledge distillation) are the main findings. They gives a better trade off between accuracy, forgetting, and efficiency than regularization based approaches or even than simple full finetuning. The main feature of PEFT replay based methods is that it keeps the performance on the early tasks very stable over time. Also, they are using a small amount of parameters from the total parameters. So, these results are the same through text classification with transformers encoders and image classification with vision transformers

Results also show backbones such as RoBERTa and ViTs can help to improve the accuracy. But, they are also obtaining the benefits from the replay and parameter efficient updates methods. Regularization based strategies like EWC, LwF, and DER++ are able to reduce partially the forgetting .Also, they show negative backward transfer. So, updates on all the parameters always generates a big memory use and training takes more time without showing advantages in the stability. So, these observations help to see replay based PEFT method as an strong baseline when class incremental is being used and there are resource constraints.

In furture work, there are many directions. First, covering other NLP tasks such as sequence labelling or question answering. Also, using larger models or even more languages. Second, there are other parameter efficient techniques like LoRA, prefix tuning, or AI3. These techniques could be combined with replay and use the same framework to the new experiments. Third, more continual learning settings are needed. For example, online or non stationary streams and variable replay budgets. Also, tasks defined by subtle domain shifts. Finally, it is possible to study moreexplainability AI techniques related to interpretation methods. Also, linking representational drift to changes in downstream performance.

# References


Ahrens, K., Abawi, F. and Wermter, S. (2021). Drill: Dynamic representations for imbalanced lifelong learning.
**URL:** *http://arxiv.org/abs/2105.08445 http://dx.doi.org/10.1007/978-3-030-86340-1*$_3$3

Boschini, M., Bonicelli, L., Buzzega, P., Porrello, A. and Calderara, S. (2023). Class-incremental continual learning into the extended der-verse, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**: 5497–5512.

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.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P., Torr, P. and Ranzato, M. (2019). Continual learning with tiny episodic memories, *Workshop on Multi-Task and Lifelong Reinforcement Learning*.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding, *Proceedings of the*

*2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186.

Ermis, B., Zappella, G., Wistuba, M., Rawal, A. and Archambeau, C. (2022). Memory efficient continual learning with transformers, *Advances in Neural Information Processing Systems* **35**: 10629–10642.

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.

Huang, Y., Zhang, Y., Chen, J., Wang, X. and Yang, D. (2021). Continual learning for text classification with information disentanglement based regularization.
**URL:** *http://arxiv.org/abs/2104.05489*

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A. et al. (2017). Overcoming catastrophic forgetting in neural networks, *Proceedings of the national academy of sciences* **114**(13): 3521–3526.

Krizhevsky, A., Hinton, G. et al. (2009). Learning multiple layers of features from tiny images. Introduces the CIFAR-10 and CIFAR-100 datasets.

Li, Z. and Hoiem, D. (2018). Learning without forgetting, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **40**: 2935–2947.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach, *arXiv preprint arXiv:1907.11692* .

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning, *Advances in neural information processing systems* **30**.

Mahabadi, R. K., Ruder, S., Dehghani, M. and Henderson, J. (2021). Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks.
**URL:** *http://arxiv.org/abs/2106.04489*

Qi, Z., Khorram, S. and Li, F. (2020). Visualizing deep networks by optimizing with integrated gradients., *AAAI*, Vol. 34, pp. 11890–11898.

Rebuffi, S. A., Kolesnikov, A., Sperl, G. and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning, *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Vol. 2017-January, Institute of Electrical and Electronics Engineers Inc., pp. 5533–5542.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, *arXiv preprint arXiv:1910.01108* .

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.

Song, Y., Wang, P., Zhu, D., Liu, T., Sui, Z. and Li, S. (2023). Repcl: Exploring effective representation for continual text classification.
**URL:** *http://arxiv.org/abs/2305.07289*

Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.-Y., Ren, X., Su, G., Perot, V., Dy, J. et al. (2022). Dualprompt: Complementary prompting for rehearsal-free continual learning, *European conference on computer vision*, Springer, pp. 631–648.

Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J. and Pfister, T. (2021). Learning to prompt for continual learning.
**URL:** *http://arxiv.org/abs/2112.08654*

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**.

Zhang, X., Zhao, J. and LeCun, Y. (2015). Character-level convolutional networks for text classification, *Advances in neural information processing systems* **28**. Introduces AG News, Yelp Review Full, Amazon Polarity, Yahoo Answers Topics, and DBpedia.

Zhuo, T., Cheng, Z., Gao, Z., Fan, H. and Kankanhalli, M. (2023). Continual learning with strong experience replay.
**URL:** *http://arxiv.org/abs/2305.13622*