Evaluation Frameworks for Catastrophic Forgetting in LLM Pretraining

CS777 M.S. R&D 1 Report

by

Ninad Parikh

25M2103

Under the guidance of

Prof. Ganesh Ramakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai 400076 (India)

November 26, 2025

Acknowledgements

I would like to thank **Prof. Ganesh Ramakrishnan** for giving me the opportunity to work on the M.S. R&D 1 Project *"Evaluation Frameworks for Catastrophic Forgetting in LLM Pretraining"* under his guidance and supervision. The support I received during this course has motivated me and encouraged me to do better.

Abstract

Large Language Models (LLMs) and deep neural networks suffer from a critical limitation known as catastrophic forgetting, where previously learned knowledge is overwritten when the model trains on new tasks or data distributions. This report reviews the fundamental causes of this phenomenon, specifically focusing on gradient conflicts during optimization. We survey comprehensive evaluation frameworks designed to capture the extent of forgetting, including foundational metrics such as Backward Transfer (BWT) [2], granular class-level metrics such as Minimal Incremental Class Accuracy (MICA) [5], and advanced Memorization and Influence tests. [3]. This review synthesizes the literature to provide a roadmap for robust continual learning in neural models.

Contents

Chapter 1

Introduction

1.1 Catastrophic Forgetting

Catastrophic forgetting is the phenomenon in which artificial neural networks tend to rapidly and drastically forget previously learned information when learning new information (see Fig. 1.1).

When a neural network is sequentially trained on multiple tasks, forgetting of earlier tasks can be expected because the network parameters are adjusted to optimize the loss on the new task, which likely pushes these parameters away from their optimum value that was found for the earlier tasks. It was appreciated early on that such forgetting would occur when incrementally training a neural network on multiple tasks, but initially it was speculated that this forgetting might be relatively mild: Hinton et al. [4] hypothesized that the many small parameter updates that work together to optimize the new task, might mostly cancel each other out in terms of their effect on previous tasks. However, this did not turn out to be the case. McCloskey and Cohen [6] and Ratcliff [7] were the first to demonstrate that sequential training of simple neural networks on disjoint sets of data results in drastic forgetting, even with only small amounts of training on the new data distribution. McCloskey and Cohen [6] also noted that this forgetting is substantially worse than that observed in humans, leading them to describe it as "catastrophic".

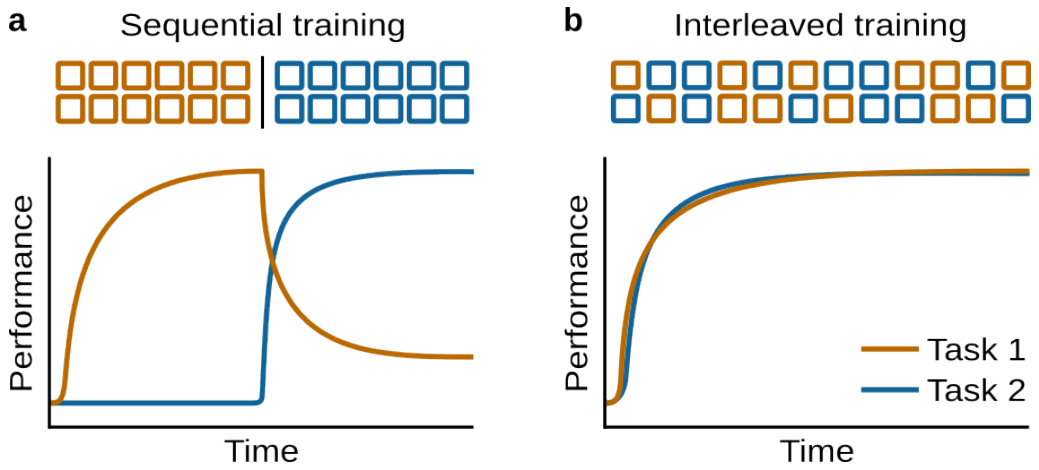

Figure 1.1: Schematic illustration of catastrophic forgetting. a, When an artificial deep neural network is sequentially trained on two tasks, it rapidly and drastically forgets the first task while training on the second one. b, Importantly, when instead trained in an interleaved fashion, the same network is able to learn both tasks, illustrating that catastrophic forgetting is not due to limited model capacity.

1.2 Why Pretraining Suffers from Forgetting

When a model learns a new task, it updates its weights to minimize the loss for that specific task. This process effectively overwrites the old knowledge or weight configurations that were optimized for previous tasks, leading to a loss of performance on earlier data.

1.2.1 Gradient Conflicts

A key insight from the multi-task learning literature is the concept of conflicting gradients [8]. Let g_i be the loss gradient with respect to the parameters of task i, and g_j be the gradient for task j. If the dot product of these gradients is negative:

$$g_i \cdot g_j < 0 \tag{1.1}$$

The gradients are said to be conflicting. Mathematically, this means that moving the parameters to improve the task i explicitly harms the task j. Standard optimizers (like SGD or Adam) simply sum these gradients or follow the current task's gradient, ignoring this destructive interference.

1.3 Need and Motivation

Neural networks trained sequentially suffer from catastrophic forgetting, where learning new tasks destructively overwrites the parameters essential for previous knowledge due to conflicting gradient updates. This limitation prevents models from adapting to new domains without losing their general reasoning capabilities. To navigate this complex optimization landscape effectively, there is a critical need for a dynamic replay buffer system. By selectively storing and re-introducing high-value examples from past data distributions, such a buffer acts as a navigational anchor, constraining gradient updates to respect prior knowledge boundaries. This approach ensures that the model balances plasticity for new tasks with stability for old ones, enabling robust continual learning without the prohibitive computational cost of retraining from scratch.

1.4 Report Outline

This paper discusses metrics to capture catastrophic forgetting, using these evaluations to identify and select high-quality data for effective model retention. The report is structured as follows: Chapter 2 details the frameworks and metrics used to capture and quantify forgetting. Chapter 3 discusses strategies to overcome these issues, focusing on gradient projection methods and replay buffer construction. Finally, the conclusions and further plans are highlighted in Chapter 4 of this report.

Chapter 2

Capturing Forgetting: Evaluation Frameworks

To effectively mitigate forgetting, we must first define rigorous metrics that go beyond simple average accuracy. We reviewed several papers that proposed various metrics for stability, plasticity, and memorization.

2.1 Foundational Continual Learning Metrics

Based on the framework proposed in "Don't forget, there is more than forgetting" [2], we define a Test-Train Accuracy Matrix R, where $R_{i,j}$ is the test accuracy on task j after the model has finished training on task i.

2.1.1 Average Accuracy (A)

The average accuracy considers the performance of the model on all tasks learned so far:

$$A = \frac{\sum_{i \geq j}^{N} R_{i,j}}{\frac{N(N+1)}{2}} \tag{2.1}$$

Although Average Accuracy (A) serves as a useful high-level indicator where a lower score signals catastrophic forgetting, it is an aggregate metric that fails to quantify the specific dynamics of knowledge transfer between tasks.

2.1.2 Backward Transfer (BWT)

Backward Transfer (BWT) measures the influence that learning a new task has on the performance of previously learned tasks [2]. This metric is essential in multi-task or data stream settings, where an agent must maintain (or improve) performance on older tasks while learning new ones throughout its lifetime.

Formally, BWT is defined as the average influence of the learning task i on all preceding tasks $j < i$. It is calculated by comparing the accuracy on task j after learning task i ($R_{i,j}$) with the accuracy on task j immediately after it was first learned ($R_{j,j}$).

The formula for Backward Transfer is given by:

$$BWT = \frac{\sum_{i=2}^{N} \sum_{j=1}^{i-1} (R_{i,j} - R_{j,j})}{\frac{N(N-1)}{2}} \tag{2.2}$$

Where:

- N is the total number of tasks.

- $R_{i,j}$ is the precision of the test in task j after the model has completed the training in task i.
- $R_{j,j}$ is the precision of the test in task j immediately after training in task j.

Interpreting BWT: Remembering and Improvement

Originally, BWT values could be positive (indicating beneficial transfer) or negative (indicating catastrophic forgetting). To distinguish these two distinct phenomena and map the metrics to a scale of $[0, 1]$, the BWT is decomposed into two clipped terms:

1. **Remembering (REM):** This metric focuses on stability by quantifying the absence of forgetting (addressing negative BWT values).

$$REM = 1 - |\min(BWT, 0)| \tag{2.3}$$

2. **Positive Backward Transfer (BWT^+):** This metric captures the ability to improve over time (addressing positive BWT values).

$$BWT^+ = \max(BWT, 0) \tag{2.4}$$

2.2 Granular Evaluation: MICA

Average accuracy metrics can mask failure modes where specific classes are completely forgotten. The **Minimal Incremental Class Accuracy (MICA)** metric [5] addresses this by tracking the worst-case performance.

$$MICA_i = \min(r_{ijk}) \tag{2.5}$$

where r_{ijk} is the accuracy of the class k after training on task i. High average accuracy combined with low MICA suggests that the model is sacrificing specific, perhaps rare, classes to maintain overall performance. This is crucial for identifying lost capabilities in code or specialized web content.

2.3 Learning Curve Area (LCA)

The Learning Curve Area [1] (LCA $\in [0, 1]$) serves as a metric to evaluate the speed with which a model learns and its performance in a few-shot scenarios.

2.3.1 Definition

First, we define the average performance of the b-shot, denoted as Z_b, where b is the mini-batch number. After training the model in all T tasks, Z_b is calculated as the average performance across all tasks after b updates.

$$Z_b = \frac{1}{T} \sum_{k=1}^{T} a_{k,b,k} \tag{2.6}$$

The LCA at β (LCA_β) is defined as the area under the convergence curve Z_b for $b \in [0, \beta]$. For discrete steps, this is calculated as the average of the Z_b values:

$$LCA_\beta = \frac{1}{\beta + 1} \int_0^\beta Z_b \, db = \frac{1}{\beta + 1} \sum_{b=0}^{\beta} Z_b \tag{2.7}$$

2.3.2 Interpretation

The LCA metric offers an intuitive way to assess learning dynamics.

- **Zero-Shot Performance:** LCA_0 represents the average 0-shot performance, which is equivalent to the Forward Transfer metric.
- **Speed of Learning:** A high LCA_β score implies that the model has both good initial performance and learns quickly.
- **Differentiation:** LCA discriminates between models that may reach the same final accuracy (Z_β) but at different speeds. A model that converges faster will have a significantly higher LCA_β than a slower model, making this metric highly relevant for evaluating learning from a few examples (small β).

2.4 Memorization and Influence

2.4.1 Context: The Long Tail Distribution

Real-world data typically follows a long-tail distribution, where a small number of examples are common, but a vast number of examples are rare. The generalization capability of a neural network often depends on its ability to memorize these rare examples [3]. This framework aims to quantify this phenomenon by measuring how much a network memorizes individual training examples and how those specific examples influence predictions on test data.

2.4.2 Definitions

Memorization

Memorization is defined as the difference in the model's accuracy in a specific training example x_i when the model is trained with that example versus when it is trained without it.

Given a data set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, a learning algorithm $\mathcal{A}$, and a specific example (x_i, y_i), memorization is calculated as:

$$\text{mem}(\mathcal{A}, S, i) = \Pr_{h \sim \mathcal{A}(S)}[h(x_i) = y_i] - \Pr_{h \sim \mathcal{A}(S \setminus \{i\})}[h(x_i) = y_i] \tag{2.8}$$

Where:

- The **first term** represents the probability that the model classifies x_i correctly when trained on the complete data set S.
- The **second term** represents the probability that the model classifies x_i correctly when trained on the data set S excluding x_i.

Influence

Influence measures the effect of a specific training example on the accuracy of a specific test example.

Given the same setup, let (x'_j, y'_j) be a test example. The influence of training example i on test example j is defined as:

$$\text{infl}(\mathcal{A}, S, i, j) = \Pr_{h \sim \mathcal{A}(S)}[h(x'_j) = y'_j] - \Pr_{h \sim \mathcal{A}(S \setminus \{i\})}[h(x'_j) = y'_j] \tag{2.9}$$

Where:

- The **first term** is the probability that the test point is correctly classified when the training set includes x_i.
- The **second term** is the probability that the test point is correctly classified when the training set excludes x_i.

2.4.3 Estimation Algorithm

To address the computational prohibitiveness of retraining the model for every leave-one-out scenario, Feldman (2020) [3] proposes an estimator based on training multiple models on random subsets of the data.

Algorithm 1 Memorization and influence value estimators

Require: Training dataset $S = ((x_1, y_1), \ldots, (x_n, y_n))$, testing dataset $S_{test} = ((x'_1, y'_1), \ldots, (x'_{n'}, y'_{n'}))$, learning algorithm $\mathcal{A}$, subset size m, number of trials t.

Sample t random subsets of $[n]$ of size m: $I_1, I_2, \ldots, I_t$.
for $k = 1$ to t **do**
    Train model h_k by running $\mathcal{A}$ on S_{I_k}.
end for
for $i = 1$ to n **do**
    $\widehat{\text{mem}}_m(\mathcal{A}, S, i) := \Pr_{k\sim[t]}[h_k(x_i) = y_i \mid i \in I_k] - \Pr_{k\sim[t]}[h_k(x_i) = y_i \mid i \notin I_k]$.
    for $j = 1$ to n' **do**
        $\widehat{\text{infl}}_m(\mathcal{A}, S, i, j) := \Pr_{k\sim[t]}[h_k(x'_j) = y'_j \mid i \in I_k] - \Pr_{k\sim[t]}[h_k(x'_j) = y'_j \mid i \notin I_k]$.
    end for
end for
return $\widehat{\text{mem}}_m(\mathcal{A}, S, i)$ for all $i \in [n]$; $\widehat{\text{infl}}_m(\mathcal{A}, S, i, j)$ for all $i \in [n], j \in [n']$.

Chapter 3

Overcoming Forgetting: Mitigation Strategies

This chapter discusses the methods reviewed to prevent or mitigate the effects described in the previous chapter.

3.1 Gradient Surgery (PCGrad)

Multi-task learning often suffers from optimization difficulties attributed to a phenomenon termed the "Tragic Triad": the co-occurrence of conflicting gradients, dominating gradients, and high curvature. To address this, Yu et al. (2020) [8] propose **Project Conflicting Gradients (PCGrad)**, an optimization approach designed to de-conflict gradients between tasks directly.

3.1.1 Gradient Conflict

Two gradients $\mathbf{g}_i$ and $\mathbf{g}_j$ of different tasks are considered to be in conflict if their directions point away from one another, formally defined by a negative cosine similarity (or negative inner product):

$$\cos \phi_{ij} < 0 \implies \mathbf{g}_i \cdot \mathbf{g}_j < 0 \tag{3.1}$$

Under these conditions, a standard gradient update typically averages these vectors, which can lead to destructive interference—improving one task at the significant expense of another .

3.1.2 The PCGrad Algorithm

The core goal of PCGrad is to modify the gradient of a task $\mathcal{T}_i$ so that it does not conflict with the gradient of any other task $\mathcal{T}_j$. This is achieved via **gradient projection**.

If a conflict is detected ($\mathbf{g}_i \cdot \mathbf{g}_j < 0$), the gradient $\mathbf{g}_i$ is projected onto the normal plane of $\mathbf{g}_j$. This operation effectively subtracts the conflicting component of $\mathbf{g}_i$ that harms the task j:

$$\mathbf{g}_i^{PC} = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_j\|^2} \mathbf{g}_j \tag{3.2}$$

If the gradients are not in conflict (non-negative cosine similarity), the original gradient remains unaltered. This procedure is repeated for all tasks in a random order to ensure constructive interaction.

Algorithm 2 PCGrad Update Rule

Require: Model parameters θ, task minibatch $\mathcal{B} = \{\mathcal{T}_k\}$

$g_k \leftarrow \nabla_\theta \mathcal{L}_k(\theta), \forall k$
$g_k^{PC} \leftarrow g_k, \forall k$
for $\mathcal{T}_i \in \mathcal{B}$ **do**
    for $\mathcal{T}_j \sim \mathcal{B} \setminus \mathcal{T}_i$ in random order **do**
        if $g_i^{PC} \cdot g_j < 0$ **then**
            $g_i^{PC} \leftarrow g_i^{PC} - \frac{g_i^{PC} \cdot g_j}{\|g_j\|^2} g_j$
        end if
    end for
end for
return update $\Delta\theta = \sum_i g_i^{PC}$

3.2 Cognitive Replay (CORE) Strategy

To address the limitations of uniform replay buffers, we examined the **Cognitive Replay (CORE)** framework proposed by Zhang et al. [9]. This framework introduces a bio-inspired approach to continual learning, drawing parallels between neural network optimization and human cognitive processes such as memory retention, recall, and review. The novelty of CORE lies in its two primary mechanisms: *Adaptive Quantity Allocation (AQA)* and *Quality-Focused Data Selection (QFDS)*.

3.2.1 Theoretical Motivation

Traditional replay methods typically allocate buffer space equally across all tasks and employ random sampling. However, this approach ignores the varying rates of forgetting between tasks and often stores redundant or low-utility samples. CORE addresses this by modeling two key human memory phenomena:

- **Cognitive Overload:** Analogous to limited buffer capacity, where new information overwrites old parameters.
- **Interference:** Modeled as gradient interference, where learning new tasks actively degrades performance on specific previous tasks.

3.2.2 Adaptive Quantity Allocation (AQA)

AQA dynamically adjusts the replay buffer quota for each task based on its specific forgetting rate, rather than using a fixed uniform distribution. This mimics the human cognitive strategy of *Targeted Recall* (prioritizing weak memories) versus *Spaced Repetition* (maintaining stable memories).

Forgetting and Interference Quantification

After training in a task sequence $\mathcal{T}_1, \ldots, \mathcal{T}_\tau$, the forgetting rate f_p for a past task p is calculated as the maximum accuracy drop:

$$f_p = \max_{i \in \{1,\ldots,\tau-1\}} (Acc_p^i - Acc_p^\tau) \tag{3.3}$$

where Acc_p^i is the accuracy on the task p after training on the task i.

Simultaneously, the potential interference i_p caused by the current task is estimated using an exponential weighting of recent accuracy drops:

$$i_p = \frac{e^{(Acc_p^{\tau-1} - Acc_p^\tau)}}{\sum_{p' \in \mathcal{P}} e^{(Acc_{p'}^{\tau-1} - Acc_{p'}^\tau)}} \tag{3.4}$$

Buffer Allocation via Attention

Using these rates, an attention score att_p is calculated for each task to determine its buffer priority:

$$att_p = -\log(1 - f_p) \tag{3.5}$$

These scores are normalized using a softmax function to derive the final buffer allocation ratios. Tasks with high attention scores (high forgetting) are categorized for *Targeted Recall* and receive larger buffer quota, while stable tasks are assigned to *Spaced Repetition* with minimal quota.

3.2.3 Quality-Focused Data Selection (QFDS)

Once buffer quota are established, QFDS ensures that the stored samples are highly representative, mirroring the human cognitive preference for deep processing of semantically rich information.

Instead of random sampling, QFDS uses the feature extractor $\mathcal{E}$ of the model. For each class, it computes a feature centroid in the latent space and iteratively selects samples whose feature embeddings are closest to this centroid. This guaranties balanced coverage and maximizes the utility of every stored example, significantly reducing redundancy compared to random selection.

Chapter 4

Summary and Future Work

4.1 Conclusion

In this report, we present a comprehensive review of the evaluation frameworks and mitigation strategies for catastrophic forgetting in neural network pretraining. We analyze a diverse set of metrics to diagnose model stability and plasticity, including Backward Transfer (BWT) [2] for global retention, Learning Curve Area (LCA) [1] for learning efficiency, and Minimal Incremental Class Accuracy (MICA) [5] for detecting granular droplets in class-specific performance. Additionally, we explore data attribution techniques such as Memorization (Exposure) and Influence [3] to understand the underlying dynamics of data retention. Finally, we examine mitigation strategies, highlighting optimization-based approaches like Project Conflicting Gradients (PCGrad) [8] as effective alternatives to standard regularization.

4.2 Future Work

Building on the insights from this review, we will focus our next steps on the development of a novel replay buffer construction technique. Taking inspiration from the Cognitive Replay (CORE) framework [9], we aim to design a strategy that goes beyond random sampling. This proposed method will take advantage of these metrics to select high-quality samples, thereby creating a dynamic self-regulating replay buffer that actively mitigates forgetting during pretraining.

Bibliography

[1] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[2] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.

[3] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

[4] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, pages 77–109. MIT Press, Cambridge, MA, USA, 1986.

[5] Mohamed Abbas Konaté, Anne-Françoise Yao, Thierry Chateau, and Pierre Bouges. Toward industrial use of continual learning: new metrics proposal for class incremental learning. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 01–07. IEEE, 2023.

[6] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[7] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[8] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

[9] Jianshu Zhang, Yankai Fu, Ziheng Peng, Dongyu Yao, and Kun He. Core: Mitigating catastrophic forgetting in continual learning through cognitive replay. *arXiv preprint arXiv:2402.01348*, 2024.