COLA: Continual Learning via Autoencoder Retrieval of Adapters

Jaya Krishna Mandivarapu
Microsoft
jmandivarapu1@microsoft.com

Abstract

Learning a set of tasks over time, also known as continual learning (CL), is one of the most challenging problems in artificial intelligence due to catastrophic forgetting. Large language models (LLMs) are often impractical to frequent re-training and continual learning , due to high cost of computational resources for training. Moreover, LLM are not suitable for continual learning as updating these models over time for acquiring new knowledge leads to overwrites existing knowledge leading to common phenomenon know as *catastrophic forgetting*. In this paper, we aim to address these concerns using a novel framework , COLA that employs an autoencoder to learn capture low-dimensional embeddings of the weights associated with various tasks. Our approach facilitates the transfer of knowledge to new tasks while preventing catastrophic forgetting, all without using data replay or a substantial set of task-specific parameters. Our approach, COLA, makes the LLM efficiently learn new tasks with minimal training, insignificant performance degradation on previous tasks, and eliminates the need for retaining earlier training data. Empirical evaluation on different datasets ranging from task oriented dialouge system to intent classsfication datasets showcases that our method not only overcomes catastrophic forgetting but also achieves significant reduction in parameter usage and memory size, across multiple tasks and outperforming the existing state of the art methods across multiple datasets.

1 Introduction

Lifelong or continual learning (CL) is one of the most challenging problems in field of AI, posing a significant hurdle in the quest for artificial general intelligence (AGI) [13, 18]. In this paradigm, a single system should continually learn to solve new tasks without forgetting previously learned tasks, while also effectively using the acquired knowledge from previous knowledge to solve new tasks. In recent times. Two primary challenges in continual learning are: (1) *preventing catastrophic forgetting*—where previously acquired knowledge is lost upon learning new tasks [30, 36] and (2) *enabling forward transfer*, which uses earlier task knowledge to speed up and improve learning on subsequent tasks. (3) *promoting backward transfer*, in which mastering new tasks enhances performance on previously learned tasks

Large Language Models have demonstrated incredible capabilities in absorbing and storing vast amounts of data within their parameters when trained on word knowledge, thus acquiring knowledge from the data. These Pre-trained LLM's has shown promising results by utilizing the knowledge encoded in them. Additionally, these models also exhibited promising results when finetuned on Knowledge Intensive Language Tasks(KILT) eg: Open Domain Question Answering, Slot Filling, Dialogue systems..etc.

One prominent application of KILT-tuned LLMs is in Task Oriented dialogue system which power modern LLM based assistants like Microsoft Coiplot, Siri and Alexa. These systems are trained to

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop (In Review): .

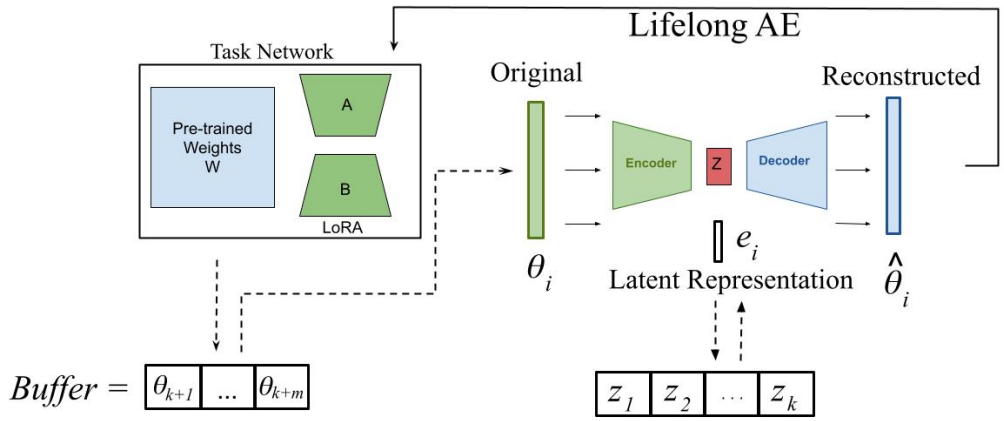

Figure 1: COLA: Our proposed framework consists of two primary components: task-specific adapter networks (TN), an optional buffer for temporarily storing the most recent m tasks, and a lifelong autoencoder (AE) designed for long-term learning. Given a new task t_k. we first train a model with parameters θ_k, We learn the optimal weights for each task and temporarily store the trained adapter in a buffer; once full, the stored adapters are encoded into the autoencoder and the buffer is cleared. Alternatively, adapters can be encoded directly without buffering. When an earlier task reoccurs (e.g., revisiting a previously learned task), we reconstruct its adapter weights by decoding the corresponding latent vector z_i from the autoencoder and load them onto the task-specific adapter. Although the latent vector z_i is asymptotically smaller than the original parameter set θ_i, the reconstructed adapter achieves performance closely matching that of the original model.

achieve specific objectives in various fields ranging from customer support agents to autonomous AI agents. Traditinally Task-oriented dialogue framework systems follows a modular pipeline: Natural Language Understanding (NLU) for understanding the user's intent, Dialogue State Tracking (DST) for tracking the current dialogue states, Dialogue Policy (DP) and Natural Language Generation (NLG) for generating the responses based on previous conversations. When these modules are trained and optimized individually, can yield high performance on individual subtasks. More researchers have begun to train task oriented dialogue systems end to end with pre-trained Large Language Models(LLM). However, these monolithic models not only demand extensive annotated data but also suffer from parameter-sharing constraints: updating the entire model to accommodate new scenarios often causes catastrophic forgetting of previously learned tasks. . Due to these limitations, extending end-to-end model to new domains and functionalities often result in high computational cost and operational costs. Therefore , the capability of the model to learn new tasks continuously without forgetting the older tasks a.k.a Continual Learning (CL) is critical for developing a efficient task oriented dialogue systems.

2 Methodology

2.1 Problem Statement

Continual learning (CL) covers a range of scenarios characterized by specific learning constraints, such as limited memory, fixed model architecture, and restricted access to previously encountered data. In this work, we consider a setting that aligns closely with realistic sustems such as classification, task-oriented dialogue systems ..etc, defined by the following constraints:

1) **Sequential acquisition.** System learns tasks in a sequential manner. Eg: New Taks (e.g., dialogue tasks, intention classification) arrive one after another; at any moment, the learner observes data from the *only one* task.
2) **Task separability.** All the task can be learned without relying on information from other tasks (e.g. domain-specific intent recognition).

3) **Task awareness.** During training the system is told which task is active, so task inference is not part of the challenge during training. During evaluation task labels are not available and task inference must be done by the system itself.

4) **No rehearsal budget.** Once training on task i is complete, System can't access task i dataset or any of the previous tasks. System inherently needs to learn new task without forgetting old tasks prevent catastrophic forgetting without replaying old data..

The above differs fundamentally from class-incremental(CIL) setting, where a *single* classification head must accommodate an ever-growing label set. Instead, our learner faces a sequence of *domain-isolated* problems and must preserve knowledge of each without revisiting past data.

Formally, each task task T_i in the CL scenarios is represented by the dataset $\mathcal{D}_i = (\mathbf{x}^{(j)}, y^{(j)})j = 1^{n_i}$, where $\mathbf{x}$ is a input and y is its output label. The CL system is sequentially exposed to datasets $(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k)$ seen exactly once during the system's lifetime. Multiple passes over $\mathcal{D}_i$ are permitted while it is resident in memory, but no element of $\mathcal{D}_i$ may be retained when the learner moves on to $\mathcal{D}_{i+1}$.

In addressing this constrained scenario, prior approaches typically fall into two categories: *Regularization* approaches aim for a *single* shared parameter set that is nudged to remain useful for old tasks, and *Expansion based* methods, which allocate a fresh parameter to each new task either by freeing old parameters or adding completely new set of parameters. The former methods offer storage efficiency at the cost of significant performance degradation over time, whereas the latter approaches achieve higher performance but suffer from linearly growing parameter storage requirements.

Contributions. We introduce a novel architectural continual learning method that strikes a balance between full-model retraining and expensive rehearsal techniques. Our contributions are threefold:

(i) We augment a fixed pretrained backbone with lightweight, task-specific adapter modules, effectively isolating new task knowledge while preserving previously acquired capabilities—without relying on raw-data rehearsal.

(ii) Upon completion of training a task, we propose to compress adapter weights using a Contractive Autoencoder (CAE), retaining only a compact latent representation and discarding the original adapter parameters. This significantly reduces storage overhead, achieving orders-of-magnitude memory savings.

(iii) The compressed latent codes enable efficient, on-demand decoding to reconstruct task-specific adapters with minimal performance loss, thereby satisfying strict no-rehearsal constraints and overcoming the prohibitive storage costs of replay buffers or full model copies.

(iv) Our latent encoding enables transfer learning by selecting and reusing the adapter with the lowest perplexity on new inputs, speeding up training and transferring knowledge across related tasks.

2.2 Problem Formulation

2.3 LoRA Adapters for Parameter-Efficient Continual Learning

Large language models (LLMs) now contain billions of parameters, making full fine-tuning prohibitive in both memory and compute. Fine-tuning a large language model(LLM) which contains billions of parameters as s backbone for every new task is unfeasible for continual learning scenarios. Instead, we employ **LoRA** (low-rank adaptation) adapter as our task-specific module of choice. LoRA posits that the *update* required to steer a pre-trained weight matrix toward a new task occupies a low-dimensional subspace. **LoRA**—Low-Rank Adaptation—addresses this by extending the network by injecting *small, trainable rank-decomposition matrices* into frozen projection layers of the Transformer [15]. When compared to traditional complete model fine-tuning, LoRA reduces the overall trainable parameters by up to four orders of magnitude and cuts GPU memory by a factor of three while achieving the similar task accuracy on GPT-2 [34], RoBERTa [24], and GPT-3 [4] benchmarks.

Low-rank update: Let's consider $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2 ... \mathbf{W}_n$ be the weights of each layer of n-layer LLM model. For a given layer $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$ be a frozen projection matrix (e.g. the query or value matrix

inside self-attention). Rather than learning a full update $\Delta\mathbf{W} \in \mathbb{R}^{d\times k}$, LoRA constrains it to a rank-r factorisation:

$$\mathbf{W} \;=\; \mathbf{W}_0 \;+\; \Delta\mathbf{W} \quad \text{with} \quad \Delta\mathbf{W} \;=\; \mathbf{B}\,\mathbf{A}, \tag{1}$$

where $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times k}$, and the rank satisfies $r \ll \min(d, k)$. During training, *only* $\mathbf{A}$ and $\mathbf{B}$ are updated, while $\mathbf{W}_0$ remains fixed.

Forward pass. Given a hidden vector $\mathbf{x}$, a LoRA-augmented linear layer computes

$$\mathbf{h} \;=\; \mathbf{W}_0\,\mathbf{x} \;+\; \mathbf{B}\big(\mathbf{A}\mathbf{x}\big), \tag{2}$$

adding at most $r(d{+}k)$ trainable parameters— typically $<2\%$ of the layer's original size.

Why LoRA suits continual learning.

(a) *Tiny footprint.* Typically using small rank values achieves the task accuracy required on par with the full fine-tuning of the model ($r = 1$–4 in our experiments). This helps in keeping the memory, compute overhead negligible. Specifically, $<2\%$ additional parameters per task, to train dozens of tasks on a single GPU.

(b) *Task Modularity.* An independent LoRA adapter is allocated per task, preventing interference without touching the backbone. This leads to acquisition of new knowledge without interference with existing knowledge. Recent work shows task-specific LoRA weights can be merged, hot-swapped, re-used at inference time for different usecases.

(c) *Compatibility with compression.* LoRA architecture perfectly suits the proposed approach of injecting the adapter into the latent space of the auto encoder. After training on a task we encode the pair $\{\mathbf{A}, \mathbf{B}\}$ with our conditional auto-encoder (2.4), store only the latent code, and discard the full low-rank weights. Thus LoRA serves as a lightweight *front-end* for task adaptation, while CAE provides long-term storage.

2.4 Auto Encoder

Our framework is not tied to any specific autoencoder design. In practice, we employ **contractive autoencoders (CAEs)** [40] because they consistently proved more stable than alternatives such as variational autoencoders. A CAE replicates the architecture of a standard autoencoder [11] but augments the loss with a contraction term that discourages large perturbations in the latent space:

$$\mathcal{L}_{\text{CAE}}(\theta) \;=\; \cos\big(\theta, \hat{\theta}\big) + \lambda\,\|J_f(\theta)\|_F^2, \tag{3}$$

where the first term is the cosine-similarity reconstruction loss and the second term is a contractive penalty. The penalty is the Frobenius norm of the Jacobian of the encoder's hidden representation with respect to each input x_i:

$$\|J_f(x)\|_F^2 \;=\; \sum_{i,j}\left(\frac{\partial h_j(\theta)}{\partial x_i}\right)^2, \tag{4}$$

with $h_j(\theta)$ denoting the activation of the j-th hidden unit. Throughout our experiments we set the regularization coefficient to $\lambda = 1 \times 10^{-4}$.

2.5 Framework Overview

Figure 1 sketches our **COLA framework** for lifelong learning without catastrophic forgetting. As shown in the fig each new task is handled in three stages—*adapt*, *compress*, and *replay on demand*(during inference)—all while keeping the backbone frozen and storing *no raw data* from previous tasks.

Stage 1: Task adaptation. When a new task T_i arrives where each task is specified as $\mathcal{D}_i$ $\mathcal{D}_i = \{(X^{(j)}, y^{(j)})\}_{j=1}^{n_i}$ which consists of n different training pairs. Our framework incorporate a rank-r LoRA matrices adapter into pre-trained frozen backbone model. We fine-tune the frozen backbone fitted with its freshly attached adapter on the current task's dataset $\mathcal{D}_i$ for task adaptaion. After

adaptation, the learned low-rank matrices $(\mathbf{A}_i, \mathbf{B}_i)$ are vectorized into a compact *adapter snapshot* $\boldsymbol{\theta}_i = \mathrm{vec}[\mathbf{A}_i; \mathbf{B}_i]$.

Stage 2: Lifelong encoding. As shown in Fig 1 for the next stage we employ a CAE for lifelong encoding. We train a *Lifelong Auto-Encoder* (LAE) to encode each adapter into a fixed-size latent vector $\mathbf{z}_i \in \mathbb{R}^m$ as shown in the Eq 5 where f indicates the encoder, g indicates the decoder of the autoencoder and overall autoencoder is optiomized using the contractive loss as shown in Eq 3.

$$\mathbf{z}_i = f_\phi(\boldsymbol{\theta}_i), \quad \hat{\boldsymbol{\theta}}_i = g_\psi(\mathbf{z}_i), \tag{5}$$

The CAE therefore learns a contractive latent manifold that captures the intrinsic geometry of LoRA updates while keeping the code dimension $m \ll |\boldsymbol{\theta}_i|$. After convergence we discard discard the *encoder* f_ϕ and store only the decoder g_ψ plus the latent vectors $\{\mathbf{z}_i\}$. This allows us to reconstruct any adapter on demand (via $g_\psi(\mathbf{z}_i)$) while reducing persistent storage by over an order of magnitude. After each task adaptation, its adapter snapshot $\boldsymbol{\theta}_i$ may be immediately sent to the LAE for encoding or temporarily held in a small FIFO buffer of size M. Whenever the buffer fills—or at scheduled intervals—we batch-encode all stored adapters via the CAE and then purge the raw weight tensors, ensuring stable auto-encoder training while smoothing out per-task encoding costs.

Stage 3: Adapter selection at inference. In a *hard* continual-learning setup, the task identity is *not* given at test time. At inference, we therefore proceed as follows:

(i) If a task-ID i is *provided*, we simply retrieve its latent code $\mathbf{z}_i$, decode it via g_ψ to reconstruct adapter $\hat{\boldsymbol{\theta}}_i$, install that adapter in the frozen backbone, and generate outputs directly.

(ii) Otherwise (no task-ID available), we must automatically choose among all K stored adapters:

(a) Decode each latent vector $\mathbf{z}_t$ into adapter weights $\hat{\boldsymbol{\theta}}_t = g_\psi(\mathbf{z}_t)$.

(b) For each adapter t, compute the model's *perplexity* on the input sequence X:

$$\mathrm{PPL}_t(X) \;=\; \exp\Big(-\tfrac{1}{|X|}\sum_{i=1}^{|X|}\log p_{\theta_t}(x_i \mid x_{<i})\Big),$$

where p_{θ_t} are the token probabilities under adapter t.

(c) Select the adapter index

$$\hat{t} \;=\; \arg\min_{1\le t\le K} \mathrm{PPL}_t(X),$$

install $\hat{\boldsymbol{\theta}}_{\hat{t}}$, and produce the response.

This scheme trades off K forward passes for a built-in confidence estimate, avoiding the need for any extra classification head (and its associated forgetting risk) while still operating under realistic "no task-label" constraints.The CAE decoder and LM forward passes for different adapters are independent, we can *batch* reconstruct all K adapters and score them in parallel, making the total added latency negligibl while only ever instantiating one adapter's weights at a time for generation.

Optional Warm-start via adapter reuse for transfer learning When a new task arrives, we first compute its perplexity under each existing adapter (as in Stage 3) and select the best-scoring adapter $\hat{t}$. We then initialize the new task's LoRA matrices $(\mathbf{A}_i, \mathbf{B}_i)$ from $\hat{\boldsymbol{\theta}}_{\hat{t}}$ rather than from random, effectively using the most relevant adapter as a *pre-trained warm start*. This transfer-learning step accelerates convergence and improves final accuracy, as evidenced by the speed-up curves in Fig 3.

Key properties. (i) *Constant storage*: the memory footprint grows with the number of *latents* not with full adapters. (ii) *No rehearsal*: the LAE operates on weights, not examples; no dialogue data are cached. (iii) *Task modularity*: the frozen backbone and per-task codes cleanly separate generic linguistic knowledge from domain-specific behaviour, enabling plug-and-play continual learning without catastrophic forgetting. (iv) *Warm-start transfer:* new tasks can initialize from the best existing adapter (via perplexity-based selection), speeding convergence and boosting performance.

3 Experiments

To validate our approach, we performed experiments on we carried out a wide range of Continual Learning experiments on a variety of datasets to to test for class incremental learning and Intent detection in Task Orient Dialogue systems, we empirically established the level of precision needed when approximating of a task network in order to achieve comparable performance on a task. Secondly, we also verified that our proposed framework can reconstruct large of task networks to test the system's ability to encode a very large number of tasks, thus validating that the proposed approach does not merely memorize task-specific networks. Finally, we assessed our method's performance on the following benchmarks.

3.1 Datasets

We experiment our method on different scenarios such as Intent Classification, dialogue state tracking (DST), Natural Language Generation(NLG) and end-to-end response generation (E2E). We performed experiments on four different datasets— Task-Master 2019 (TM19) [6], Task-Master 2020 (TM20) [6], Schema Guided Dialogue (SGD) [35] and MultiWoZ [46, 12] following the methodology defined in Section 2. Below, we first present our dataset and implementation details, then discuss our results.

We experiment our methods with the following datasets.

CLINC150 This dataset [20] is a multi-domain dataset that contains 23,700 utterances used for evaluation of intent classification and out of scope prediction over wide range of intents. Overall this dataset cover 150 intent classes over 10 domains.
MultiWOZ This dataset [5] craved using Wizard-of-Oz (WOZ) setup. MultiWOZ compraises of multi-turn dialogues,which is widely used to evaluate the task oriented dialogue(TOD) systems. Datasets contains of domains like restaurant,..etc. There are two versions of this dataset: Initial version proposed by [46] and later [12] corrected version the addressing the errors and inconsistencies. We also use task oriented datasets such as Task-Master 2019 (TM19) [6], Task-Master 2020 (TM20) [6], Schema Guided Dialogue (SGD) [35]
Task Master: This datasets consists of corpus of 55,000 spoken and written task-oriented dialogues in over a different domains. Task-Master 2019 (TM19) [6] contains 6 domains eg: restaurants, food ordering, moviews ..etc, Task-Master 2020 (TM20) [6] contains 23,789 movie ticketing dialogues which span over domains such as deciding on theater, time, movie name, number of tickets, and date, or opt out of the transaction..etc.
Schema-Guided Dialogue (SGD) This dataset[35] contains over 20k annotated examples spanning over 20 domains such as s banks, events, media, calendar..etc. These annotated examples are the interactions between the Virtual assistant and human over services and API's.

We also utilize a combines dataset Natural Language understanding that is a relabeled and aggregated version of three large NLU corpuses: CLINC150 [20] , Task-Master 2020 (TM20) [6], Schema Guided Dialogue (SGD) [35] and MultiWOZ [5].

Name	Train	Valid	Test	Dom.	Intent	Turns
TM19	4,403	551	553	6	112	19.97
TM20	13,839	1,731	1,734	7	128	16.92
MWoZ	7,906	1,000	1,000	5	15	13.93
SGD	5,278	761	1,531	19	43	14.71
CLINIC150	10,525	3,080	5,470	-	10	-
Total	41,951	7,123	10,288	37	290	16.23

Table 1: Main datasets statistics.

3.2 Implementaion Details

3.2.1 State of Art Comparison and Metrics

For experiments on Task-Master 2019, Task-Master 2020 (TM20) [6], Schema Guided Dialogue (SGD) [35] and MultiWOZ [5] we used a decoder-only model GPT-2 where froze the model parame-

ters and updated only the adapter parameters. For adapter, we used bottleneck sizes of bottleneck size b between 10, 50, 100, 200. We used an auto encoder with latent vector of size 50.

Metrics: For *Intent Recognition*: We measure the precision of intent prediction by comparing the predicted intent of the model with the gold standard label. For *Dialogue State Tracking (DST)* the performance is quantified using Joint Goal Accuracy (JGA) as define as the percentage of dialogue turns where all slot-value pairs match the gold state. and for *Natural Language Generation (NLG)*: Quality is assessed with the BLEU metric [33] along with the slot error rate (SER [26]), calculated as the fraction of required slot values that are missing or incorrect in the generated response.

Baselines: We compare our method across various existing approaches.

- **Adapter-style [26]**: inserts lightweight adapter modules into the base model; each task trains its own adapter, enabling modular task-specific learning without overwriting shared model weights.
- **Replay (GEM) [10]**: fine-tunes the entire model while storing and replaying real samples from prior tasks via a memory buffer; uses gradient projection to avoid catastrophic forgetting.
- **EWC [19]**: fine-tunes the model with an additional regularization term that penalizes changes to parameters important for previous tasks, based on the Fisher Information Matrix.
- **L2 Regularization [14]**: constrains the model by penalizing deviation from previously learned weights using a simple L2 loss, encouraging retention of earlier knowledge.
- **Vanilla FT**: sequentially fine-tunes the entire model on each task without any regularization or memory, typically resulting in catastrophic forgetting.

3.2.2 Results

Table 3.2.2 presents a performance comparison of our method against various continual learning methods across multiple CL datasets. First, we trained a LoRA adapter attached to pre-trained gpt2 network, in which only LoRA parameters are updated for learning the task while rest of the network remains constant. The LoRA network is flatted and splitted to fed into our Our AE which consists of two connected layers with 100,000, 50 parameters respectively. Thus, our latent vectors were of size 50. Each task network can be ingested into auto encoder in a sequential fashion.

We report results from average of five random runs with different task orders for each run on the CL benchmark. In fairness, we used the same rank of the LoRA rank adapter in each corresponding comparison experiment that utilize adapter-style learning. Across all tasks on different datasets of the standard CL benchmark, Our method consistently outperforms previous methods by a consistent margin and on par with task-specific models trained independently, and highlighting the effectiveness of our approach. Our methods also significantly surpass several state-of-the-art continual learning approaches, including EWC, L2, and AGEM, LAMOL and replay based methods. Unlike these baselines, our methods retain knowledge of previous tasks while maintaining the ability to learn new ones effectively.

We evaluated the average task performance relative to changes in model parameters generated by the auto encoder, quantified using reconstruction score. As shown in Fig 2 results reveal a strong correlation: as reconstruction similarity score decreases, performance tends to decrease. The baseline accuracy is shown as a dotted red line, marks the point where the which model performance closely matches that of the original. Therefore, we use a threshold of 0.99 as the stopping criterion in our following experiments, unless otherwise specified.

3.3 Continual Learning Transfer Validation

To validate the effectiveness of our continual learning approach, we conducted a controlled experiment examining knowledge transfer between sequential tasks using the CLINC-OOS intent classification dataset. The experiment consisted of three phases: (1) training a RoBERTa-base model with LoRA adapters on Task 1 (achieving 95.33% accuracy), (2) training a fresh model on Task 2 from scratch, and (3) initializing Task 2 training with the pre-trained Task 1 model weights. Our results demonstrate significant evidence for positive knowledge transfer in continual learning scenarios. The model pre-trained on Task 1 achieved 95.00% accuracy on Task 2, representing a substantial 17.33 percentage

Method	Accuracy↑	JGA↑	EER↓	BLEU↑
VANILLA	5.12	8.91	50.76	7.48
L2 [14]	4.81	7.21	60.88	5.8
EWC [19]	4.85	9.56	61.33	5.56
AGEM [10]	39.14	12.73	67.99	4.64
LAMOL [43]	12.59	8.45	71.12	3.2
REPLAY [38]	82.38	30.32	**16.12**	**17.9**
Ours	**89.87**	**34.86**	32.58	16.26
MULTI	93.45	47.3	11.46	22.51

Table 2: E2E results in terms of Intent accuracy, Joint Goal Accuracy (JGA), Slot Error Rate (EER), and BLEU. **+Param** indicates additional parameters per task, and **Mem** indicates episodic memory per task.

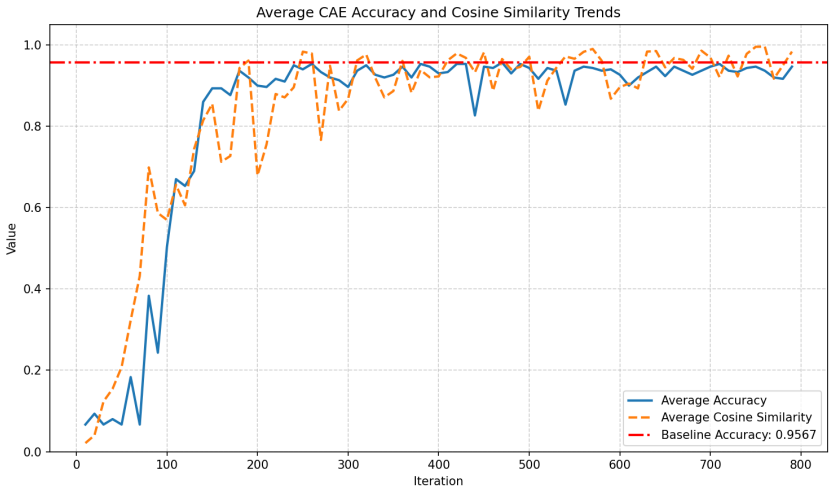

Figure 2: COLA: Avg Accuracy vs Baseline vs Reconstruction Score

point improvement over the fresh model trained from scratch (77.67% accuracy). This improvement indicates that the learned representations from Task 1 contain transferable knowledge that accelerates learning and enhances performance on related downstream tasks.

The experimental validation confirms that continual learning through parameter-efficient fine-tuning enables effective knowledge reuse across sequential tasks. The pre-trained model not only achieved superior final performance but also demonstrated faster convergence, reaching high accuracy levels within the same training epochs as the baseline. This finding supports the fundamental hypothesis that neural networks can accumulate and leverage knowledge from previous tasks to improve learning efficiency on new but related problems. The success of this transfer learning approach, particularly in the context of intent classification where tasks share underlying linguistic patterns and semantic structures, provides empirical evidence for the viability of continual learning systems in practical NLP applications.

3.4 Parameter and Memory Overhead in Continual Learning Approaches

Table 3.4 presents a comparative analysis of various continual learning (CL) strategies in terms of their storage overhead during both training and deployment. The comparison includes replay-based, regularization-based, and modular adaptation approaches.

Our method, CAE Replay, stands out as a highly efficient solution. During training, it requires storing only the decoder parameters ($n\phi$) in addition to the shared base model (Θ), avoiding the need to retain large memory buffers or multiple full model copies. At deployment time, the advantage becomes even more pronounced: unlike replay-based methods such as Replay or GEM, which must retain the

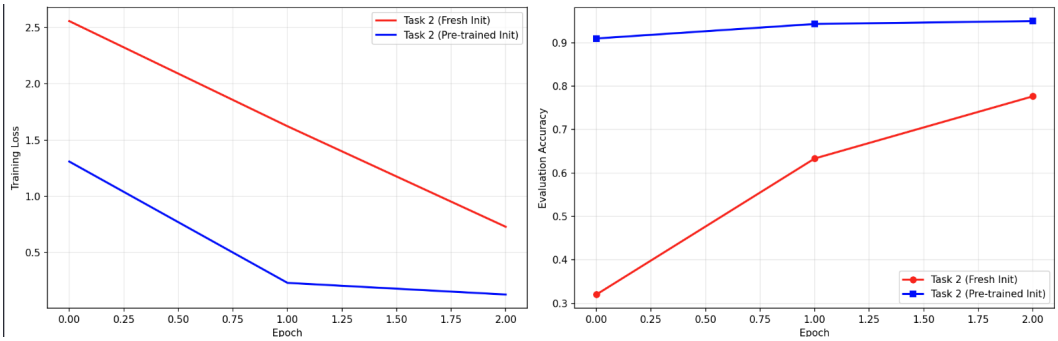

Figure 3: COLA: Avg Accuracy vs Baseline vs Reconstruction Score

Method	Train Overhead	Deploy Overhead	Notes
CAE Replay	$\Theta + n\phi$	$n\phi$	✓ Lightweight
Replay / GEM	$\Theta + n\|M\|$	$\Theta + n\|M\|$	✗ High memory
Adapter-style	$\Theta + n\|\mu\|$	$n\|\mu\|$	✓ Modular
EWC	Θ	Θ	▲ Forgetting?
L2 Reg.	Θ	Θ	▲ Forgetting?
Vanilla	Θ	Θ	✗ No memory

Table 3: Comparison of storage overhead across continual learning methods.

full memory buffer ($n|M|$), and adapter-based methods that store task-specific modules ($n|\mu|$), CAE Replay only retains compact task-specific decoders.

This significantly reduces storage cost without compromising task-specific adaptability. The method is especially suitable for deployment scenarios with limited memory capacity, such as edge devices or large-scale distributed systems, where lightweight task modules are critical. Furthermore, unlike regularization-based methods (e.g., EWC or L2), which are storage-efficient but prone to catastrophic forgetting, CAE Replay retains strong task performance by leveraging its generative memory mechanism through compact decoders.

4 Related Work

Continual Learning a.k.a Lifelong Learning aims to solve the problem of achieving good performance on alot of numbers of tasks that are arriving sequentially without catastrophic forgetting [19, 30]. Based on the nature of the tasks, Continual learning can be broadly divided into three categories: Class-Incremental Learning (CIL), Domain-Incremental Learning (DIL), Task Incremental Learning (TIL).

- **Class-Incremental Learning (CIL)** The model must continually expand its set of recognizable classes and correctly classify inputs among all seen classes.
- **Domain-Incremental Learning (DIL)** Methods that fine-tune large language models on a stream of domain-specific instructions, improving their ability to perform tasks within a particular subject area.
- **Task-Incremental Learning (TIL)** Approaches that sequentially fine-tune a model on each new task—knowing task boundaries—so that it acquires the capability to solve each distinct task as it arrives.

Depending on how each task's knowledge is stored and subsequently leveraged as new tasks arrive, task-incremental methods can be divided into three categories. 1) *Regularization-based* : where regularization based constraints are introduced on model parameters by modifying the loss function to overcome catastrophic forgetting while learning new tasks. Some of the methods include EWC ..etc [9, 1, 19] 2. *Reply-Based methods* where data from previous tasks either in the form of exemplars or buffer or complete dataset are explicitly stored and revisited to learn new things. Key appraoches

include Experience Replay[31], iCarl [37] and some othe methods include [2, 7, 32, 3, 41, 25, 44, 23]. 3.*Architecture-based Networks* where model architecture is evolute as the model training goes on while learning new task. This happens either expanding the network to include new knowledge [27, 39, 45, 29, 42, 16, 28, 8]

Several recent studies have shown that fine-tuning only a small portion of a model's parameters can match the performance of full-model updates [17, 21, 22]. Although most of this work targets single-task learning, a few efforts have begun to apply parameter-efficient approaches within continual learning. For example,[26] introduce AdapterCL, which assigns a unique adapter module to each new task.However, these approaches either demand the continual growth of adapter modules or maintain a fixed adapter architecture, which in turn limits their capacity to integrate new information.

5 Conclusion

In this work we reframed continual learning, and more generally task-oriented dialogue learning, as a *generative* problem solvable by a frozen language-model backbone equipped with LoRA adapters. Our methodology couples a pre-trained base model with MIXADAPTER blocks, allowing the system to acquire new skills sequentially while leaving the backbone untouched. Beyond simplifying multi-task training, adapter isolation highlights the classic trade-off among parameter count, replay memory, and model capacity: storing knowledge in many small bottlenecks removes the need for large episodic buffers yet still preserves prior competence. This finding underscores a broader *no-free-lunch* principle in lifelong dialogue systems: memory efficiency, model plasticity, and final performance must be balanced rather than optimized in isolation. Our experiments demonstrate that the proposed framework *efficiently acquires and retains a large number of tasks* without catastrophic forgetting, validating its effectiveness for real-world continual dialogue scenarios.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[2] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

[3] Lorenzo Bonicelli, Matteo Boschini, Angelo Porrello, Concetto Spampinato, and Simone Calderara. On the effectiveness of lipschitz-driven rehearsal in continual learning. *Advances in Neural Information Processing Systems*, 35:31886–31901, 2022.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*, 2018.

[6] Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Andy Cedilnik, and Kyu-Young Kim. Taskmaster-1: Toward a realistic and diverse dialog dataset. *arXiv preprint arXiv:1909.05358*, 2019.

[7] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. *arXiv preprint arXiv:2104.05025*, 2021.

[8] Blake Camp, Jaya Krishna Mandivarapu, and Rolando Estrada. Continual learning with deep artificial neurons, 2020.

[9] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018.

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

[11] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[12] Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tur. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*, 2019.

[13] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[14] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[15] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[16] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in neural information processing systems*, 32, 2019.

[17] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021.

[18] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[20] Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*, 2019.

[21] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[22] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[23] Yan-Shuo Liang and Wu-Jun Li. Loss decoupling for task-agnostic continual learning. *Advances in Neural Information Processing Systems*, 36:11151–11167, 2023.

[24] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[25] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continuum learning. In *NIPS*, 2017.

[26] Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, and Zhiguang Wang. Continual learning in task-oriented dialogue systems. *arXiv preprint arXiv:2012.15504*, 2020.

[27] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[28] Jaya Krishna Mandivarapu, Blake Camp, and Rolando Estrada. Self-net: Lifelong learning via continual self-modeling. *Frontiers in Artificial Intelligence*, Volume 3 - 2020, 2020.

[29] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *CoRR*, abs/1802.01569, 2018.

[30] 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.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *ArXiv e-prints*, December 2013.

[32] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[33] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[35] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8689–8696, 2020.

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

[37] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *CoRR*, abs/1611.07725, 2016.

[38] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[39] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive Neural Networks. *ArXiv e-prints*, June 2016.

[40] Rifai Salah, P Vincent, X Muller, X Gloro, and Y Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proc. of the 28th International Conference on Machine Learning*, pages 833–840, 2011.

[41] Fahad Sarfraz, Elahe Arani, and Bahram Zonooz. Error sensitivity modulation based experience replay: Mitigating abrupt representation drift in continual learning. *arXiv preprint arXiv:2302.11344*, 2023.

[42] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018.

[43] Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329*, 2019.

[44] Zifeng Wang, Zheng Zhan, Yifan Gong, Yucai Shao, Stratis Ioannidis, Yanzhi Wang, and Jennifer Dy. Dualhsic: Hsic-bottleneck and alignment for continual learning. In *International Conference on Machine Learning*, pages 36578–36592. PMLR, 2023.

[45] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[46] Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. *arXiv preprint arXiv:2007.12720*, 2020.