In-context Continual Learning Assisted by an External Continual Learner

Saleh Momeni [1], Sahisnu Mazumder [2], Zixuan Ke [3], Bing Liu [1]
[1] Department of Computer Science, University of Illinois Chicago, USA
[2] Intel Labs, USA [3] Salesforce AI Research, USA
smomen3@uic.edu, sahisnumazumder@gmail.com,
zixuan.ke@salesforce.com, liub@uic.edu

Abstract

Existing continual learning (CL) methods mainly rely on fine-tuning or adapting large language models (LLMs). They still suffer from *catastrophic forgetting* (CF). Little work has been done to exploit in-context learning (ICL) to leverage the extensive knowledge within LLMs for CL *without* updating any parameters. However, incrementally learning each new task in ICL necessitates adding training examples from each class of the task to the prompt, which hampers scalability as the prompt length increases. This issue not only leads to excessively long prompts that exceed the input token limit of the underlying LLM but also degrades the model's performance due to the overextended context. To address this, we introduce InCA, a novel approach that integrates an *external continual learner* (ECL) with ICL to enable scalable CL without CF. The ECL is built incrementally to pre-select a small subset of likely classes for each test instance. By restricting the ICL prompt to only these selected classes, InCA prevents prompt lengths from becoming excessively long, while maintaining high performance. Experimental results demonstrate that InCA significantly outperforms existing CL baselines, achieving substantial performance gains.[1]

1 Introduction

Continual learning (CL) aims to enable models to learn a sequence of tasks incrementally (Chen and Liu, 2018; De Lange et al., 2021). CL is typically categorized into three main settings: *task-incremental learning*, *class-incremental learning* (CIL), and *domain-incremental learning* (Van de Ven and Tolias, 2019). In this paper, we focus on the CIL setting (Rebuffi et al., 2017), where each task has a set of distinctive classes, and a single model is developed to handle all tasks and classes. At test time, no task information is provided for each test instance. This differs from task-incremental learning, which provides the task-id for each test instance, making classification much easier. CIL requires a unified model that can distinguish all classes seen thus far. In domain-incremental learning, all tasks have the same classes but are from different domains.

There are two key challenges in CIL. **(1) *Catastrophic forgetting*** (CF), which refers to the performance deterioration of earlier tasks due to parameter updates in learning new tasks (McCloskey and Cohen, 1989). **(2) *Inter-task class separation*** (ICS), which refers to the phenomenon that without accessing the previous task data, the learning of a new task has difficulty in establishing decision boundaries between the new and old classes (Kim et al., 2023). Although the CL community has studied CF extensively, the challenge of ICS has only been identified recently in (Kim et al., 2022). Both challenges disappear in ***in-context CIL*** with LLMs. A simple method to apply in-context learning to CIL is to incrementally add few-shot training examples for each new class to the in-context prompt. This prompt includes examples from all classes encountered so far along with instructions for classification. Since the LLM parameters remain unchanged, CF is inherently avoided, and ICS is addressed by encompassing all classes and their examples within the same prompt.

Unfortunately, this approach is not scalable for CIL because the prompt length rapidly increases with each new task or class added, quickly exceeding the token limits of LLMs. Although summarizing the training examples can increase the number of classes that can be learned (i.e., included in the prompt), the underlying scalability problem persists. Moreover, including excessive and often irrelevant information from various classes leads to significant performance degradation (see Section 6). Even with the recently introduced long-context LLMs (Chen et al., 2024; Reid et al., 2024), our experiments demonstrate that the performance degradation persists despite the increased token capacity.

In this paper, we introduce InCA (**In**-context **C**ontinual Learning **A**ssisted by an External Continual Learner), a novel method that overcomes the

[1]Published as a conference paper at COLING 2025.

scalability and performance limitations of in-context CIL while retaining the advantages of in-context learning – specifically, avoiding CF and ICS problems. InCA leverages an *external continual learner* (ECL) that both benefits from and enhances the LLM's in-context learning capabilities. The ECL aims to reduce the number of candidate classes to a small set of k classes that are most likely to include the correct class. For each input instance, we first prompt the LLM to generate a list of *tags*–descriptive topics or keywords that capture the essential semantics of the input text (see Figure 1 for an illustrative example). Each class in the dataset is represented by a Gaussian distribution over the embeddings of these tags, characterized by a *mean vector* and a *shared covariance matrix*. The ECL then computes the Mahalanobis distance (De Maesschalck et al., 2000) between the input's tag embeddings and each class distribution to identify the top k most similar classes. These selected classes are then used to construct an in-context learning prompt, efficiently managing the token limit while removing irrelevant information.[2]

Unlike traditional CL methods, our ECL requires no additional training – it only incrementally accumulates and updates class means and a shared covariance matrix derived from the embeddings of the tags generated by the LLM. This approach inherently avoids CF. Moreover, representing each class with a Gaussian distribution addresses the ICS problem, as different classes are naturally distinguished by their statistical distributions. While the ECL alone (e.g., performing top-1 classification based on Mahalanobis distance) can be applied to CIL, its standalone accuracy is limited. However, when integrated with the LLM's in-context learning, InCA significantly improves performance, as demonstrated in our experiments (see Section 6). This approach effectively balances scalability and accuracy, making in-context CIL feasible and efficient.

To summarize, our contributions are as follows:

1. We introduce the novel paradigm of in-context CIL, which, to the best of our knowledge, has not been previously studied.

2. We propose InCA, a new method that addresses token limit constraints and performance degradation caused by overextended context in in-context CIL.

3. Our method surpasses existing state-of-the-art CIL baselines, achieving significant performance improvements across different benchmark datasets.

2 Related Works

There is a large body of literature on continual learning. The main focus is on dealing with CF. Existing techniques can be broadly classified into a few categories. (1) *Regularization*, which uses a regularizer to ensure that important network parameters from previous tasks are minimally altered when learning new tasks, thereby reducing CF (Li et al., 2022; Liu et al., 2019). (2) *Replay*, which stores some training samples from previous tasks. When learning a new task, the model is trained using both the new task data and the stored replay data to mitigate CF (Liu et al., 2021a; Qin et al., 2022; Huang et al., 2021). Some replay methods do not store actual data but learn data generators to generate data similar to those from previous tasks (Shin et al., 2017; He and Jaeger, 2018). (3) *Architectural-based*, which encompasses various methods aimed at managing CF through structural modifications. Some techniques expand the network's capacity as new tasks are learned (Wang et al., 2022a; Yan et al., 2021; Qin et al., 2023). Some do *parameter isolation*, which trains sub-networks for each task by using masks to prevent updates to critical parameters or neurons from previous tasks, or by ensuring that new task parameters are orthogonal to those of prior tasks (Ke et al., 2021a, 2023; Konishi et al., 2023; Serra et al., 2018; Gururangan et al., 2022; Zhu et al., 2022; Geng et al., 2021; Lin et al., 2022; Liu et al., 2023).

Moreover, some methods incorporate parameter-efficient fine-tuning (PEFT) techniques such as low-rank adaptation (LoRA) (Hu et al., 2021) and prompt-tuning to allocate task-specific parameters for each new task (Razdaibiedina et al., 2023; Wang et al., 2022b, 2024b). These systems often implement various mechanisms to predict the task-id, which is essential for selecting the appropriate model for CIL. They may utilize a separate network, entropy, or out-of-distribution detection to predict the task-id (Rajasegaran et al., 2020; Abati et al., 2020; Kim et al., 2023). Our work differs from these approaches as it does not require task-id prediction. While the aforementioned methods train a different model for each task and rely on task-ids, our approach allows for adding one class at a time with no task-id prediction. Our ECL directly predicts the most probable classes.

[2] Our experiments demonstrate that the ECL achieves high top-k recall, ensuring that the correct class is almost always included in the top k classes to be used in the final in-context learning prompt.

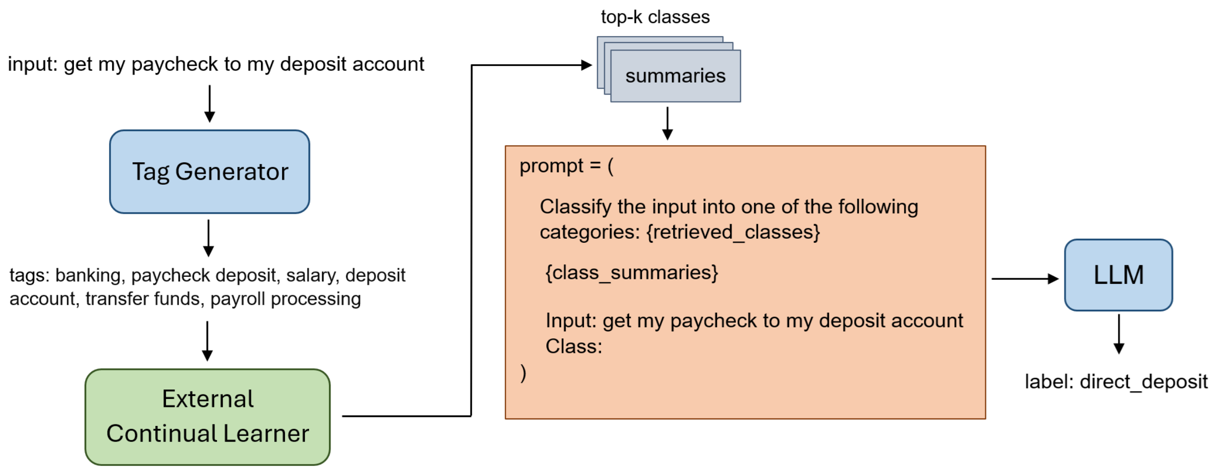
top-k classes
summaries
input: get my paycheck to my deposit account
Tag Generator
tags: banking, paycheck deposit, salary, deposit account, transfer funds, payroll processing
External Continual Learner
prompt = (
Classify the input into one of the following categories: {retrieved_classes}
{class_summaries}
Input: get my paycheck to my deposit account
Class:
)
LLM
label: direct_deposit

Figure 1: Overview of the InCA framework. The diagram depicts the stages of generating semantic tags for the input, identifying the most similar classes via the ECL, and constructing the prediction prompt with class summaries, which together enables efficient in-context continual learning without retaining any training data.

In the field of NLP, CL has been employed to address a variety of problems, including text classification (Chuang et al., 2020), sentiment analysis (Ke et al., 2021a), topic modeling (Gupta et al., 2020), slot filling (Shen et al., 2019), question answering (Greco et al., 2019), language learning (Li et al., 2019; Liang et al., 2024; Zhao et al., 2024b), and the pre-training of language models (Ke et al., 2023; Qin et al., 2022). Pre-trained models are commonly utilized in most NLP-related continual learning tasks, serving as a standard practice (Ke et al., 2021b; Wang et al., 2024c). For further insights and a comprehensive overview, refer to surveys (Ke and Liu, 2022; Wang et al., 2024a).

Our approach differs from the aforementioned methods that adapt or fine-tune pre-trained language models, as we primarily leverage in-context learning for CIL. While advancements in LLMs have improved few-shot and instruction-based prompting (Wei et al., 2022; Yao et al., 2024; Hao et al., 2023), they fail to address CL challenges, such as growing prompt sizes that quickly exceed token limits. Moreover, even with long-context LLMs (Reid et al., 2024; Dubey et al., 2024), extended prompts containing excessive and often irrelevant information can lead to significant performance degradation. InCA overcomes these issues by using an external continual learner that can be updated without CF.

InCA bears some resemblance to retrieval-augmented generation (RAG) (Zhao et al., 2024a), which uses a retriever to gather information to provide domain-specific knowledge for enhancing content generation. However, InCA is fundamentally different, as our ECL is a coarse-grained classifier that tries to identify the most similar classes rather than retrieving domain- or task-specific content. Additionally, due to the incremental nature of CIL, our ECL must be built incrementally and handle CF and ICS without storing data from previous tasks – challenges that retrievers do not encounter.

3 Problem Formulation

We study class-incremental learning in the text classification domain. CIL involves learning a sequence of tasks arriving sequentially (Kim et al., 2023). Let B be the number of tasks encountered so far. Each task b ($1 \leq b \leq B$) is associated with a training dataset $\mathcal{D}_b = \{(x_b^{(i)}, y_b^{(i)})\}_{i=1}^{n_b}$, where n_b is the total number of instances in $\mathcal{D}_b$, $x_b^{(i)}$ denotes an input (text) instance, and $y_b^{(i)}$ is its corresponding class label. Let $\mathbf{Y}_b$ be the set of classes belonging to b (i.e., the set of all classes in $\mathcal{D}_b$). For any two tasks b and b', their corresponding class sets are disjoint ($\mathbf{Y}_b \cap \mathbf{Y}_{b'} = \emptyset$ for $b \neq b'$). The overall class set for all B tasks is defined as $\bigcup_{b=1}^{B} \mathbf{Y}_b = \mathbf{Y}$. The goal is to construct a unified predictive function $f : \mathbf{X} \rightarrow \mathbf{Y}$ capable of classifying any given test instance x across all tasks/classes seen so far, despite the restriction that no data from previous tasks are retained during training, i.e., **replay-free**.

4 Proposed InCA Method

This section presents InCA (In-context Continual Learning Assisted by an External Continual Learner), a framework designed to address the challenges of CIL by leveraging the in-context learning capabilities of LLMs. InCA has three main stages: (1) **tag generation**, where semantic tags are extracted from the input text using the LLM (Section 4.1); (2) **external continual learning**, which identifies the top k most probable classes based on the generated tags through Gaussian class modeling and Mahalanobis distance scoring (Section 4.2); and (3) **in-context learning with class summaries**, where the LLM predicts the final class label for the input text using summaries of the top k candidate classes (Section 4.3). Figure 1 depicts the overall framework.

4.1 Tag Generation

To capture the essential semantic information from an input text x, we generate a list of tags that include topics, keywords, important entities, and other relevant elements. For example, a customer's banking query processed by our framework (see Figure 1) might generate tags such as "banking" or "paycheck deposit" while omitting less pertinent information. Additionally, the tags are automatically extended to include related terms that commonly appear in similar contexts. For instance, the tag "paycheck deposit" may be extended to include terms like "transfer funds" or "payroll processing". Tags are generated by prompting the LLM to produce both primary tags and related terms. The specific prompt used for tag generation is detailed in Appendix A.2.

4.2 External Continual Learner

The ECL leverages the generated tags to identify the k most probable classes for a given input, thereby filtering out the irrelevant context. As mentioned earlier, the ECL operates by accumulating statistics without additional training and thus, inherently avoids CF.

Gaussian Class Representation: Each class is modeled as a Gaussian distribution, with a mean vector and a shared covariance matrix. This representation helps mitigate the ICS problem by allowing classes to have independent distributions. However, since the covariance matrix has high dimensionality, storing a separate covariance matrix for each class would result in excessive space consumption. To address this, we assume that all classes share the same covariance matrix, drastically reducing the space required.

Let $\mathcal{T}_j = [t_{1,j}, t_{2,j} \ldots, t_{R,j}]$ be the list of all tags generated by the LLM for class j, where R is the total number of tags generated from all training instances of class j. We employ the widely-used Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) model to encode each tag $t_{r,j} \in \mathcal{T}_j$ into a h-dimensional embedding vector $z_{r,j} \in \mathbb{R}^h$. The mean vector $\mu_j \in \mathbb{R}^h$ for class j is computed as the average of all its tag embeddings:

$$\mu_j = \frac{1}{R} \sum_{r=1}^{R} z_{r,j}$$

The shared covariance matrix $\Sigma \in \mathbb{R}^{h \times h}$ is updated incrementally as new classes are introduced. The contribution of class j to the shared covariance matrix, denoted as Δ_j, is based on the deviations of its tag embeddings from the mean (Park et al., 2018):

$$\Delta_j = \frac{1}{R} \sum_{r=1}^{R} (z_{r,j} - \mu_j)(z_{r,j} - \mu_j)^T$$

The overall shared covariance matrix is updated after each new class is processed:

$$\Sigma_j = \frac{(j-1)\Sigma_{j-1} + \Delta_j}{j},$$

where Σ_j denotes the shared covariance matrix after processing class j, assuming that classes $\{1, 2, \ldots, j-1\}$ have been previously learned.

Mahalanobis Distance Scoring: For each test instance, the ECL uses the Mahalanobis distance (De Maesschalck et al., 2000) to select the top k most similar classes. Let $\{z_i\}_{i=1}^m$ be the set of tag embeddings generated for the input instance x. The Mahalanobis distance between an embedding z_i and the Gaussian distribution for class j is computed as:

$$d(z_i, \mu_j, \Sigma) = \sqrt{(z_i - \mu_j)^T \Sigma^{-1} (z_i - \mu_j)}$$

Here, Σ represents the shared covariance matrix updated up to the current point when the inference is performed. The overall distance of the test instance x from the class is the average Mahalanobis distance over all tag embeddings:

$$d(x, \mu_j, \Sigma) = \frac{1}{m} \sum_{i=1}^{m} d(z_i, \mu_j, \Sigma)$$

The top k classes with the smallest Mahalanobis distances are selected for the final prediction step.

4.3 In-context Learning with Class Summaries

Once the top k candidate classes are identified by the ECL, in-context learning is applied using **class summaries** to determine the final prediction. Storing and using too many training examples for in-context learning would be impractical and inefficient for continual learning. Instead, for each class, we generate a summary at the time of its introduction, serving as a compact representation of the class.

Generating Class Summaries: The summary for each class is generated by prompting the LLM using a small subset of randomly selected training examples of the class. The prompt used for generating the summary is given in Appendix A.1. Each summary captures the essential characteristics or information of the class, allowing for efficient in-context learning without storing lots of examples.

Prediction with In-context Learning: During prediction, the test instance is concatenated with the summaries of the top k classes to form a single prompt. The LLM processes this prompt and predicts the class label based on the context provided. The prompt format for this prediction stage is detailed in Appendix A.3.

To summarize, InCA stores only a mean embedding vector for each class and a shared covariance matrix for all classes encountered so far. Thus, the amount of information saved in the CIL process is very small. The whole process involves **no training** and it is **replay-free**, i.e., no previous task data is stored to help deal with CF. Moreover, as explained earlier, it avoids both the CF and ICS problems that have plagued the existing CIL techniques.

5 Experiment Setup

In this section, we describe the datasets, baselines, implementation details, and evaluation metrics.

Datasets: We utilize four datasets for our experiments: CLINC (Larson et al., 2019), Banking (Casanueva et al., 2020), HWU (Liu et al., 2021b), and DBpedia (Auer et al., 2007). The intent classification datasets – CLINC, Banking, and HWU comprise of 150, 77, and 64 classes, respectively. DBpedia is a topic classification dataset with 70 classes. For the train/test splits, we allocate 10k/750 samples for CLINC, 10k/1k samples for Banking, 9k/1k samples for HWU, and 10k/1k samples for DBpedia.[3] InCA can incrementally learn each class one by one. For baselines, we adhere to the standard CIL protocol by splitting the classes into disjoint tasks, each composed of a subset of classes. Multiple runs with different task splits are conducted, and the accuracy values are averaged to minimize the influence of any specific task split on overall performance.

Baselines: We evaluate our proposed method against several baselines. The **Vanilla** baseline sequentially fine-tunes the LLM on each task without any specific mechanism to mitigate CF. **EWC** (Kirkpatrick et al., 2017) is a regularization-based technique that mitigates CF by preserving important parameters for previous tasks through a quadratic penalty. **L2P** (Wang et al., 2022b) freezes the LLM's parameters and learns a set of trainable prompts to guide the model during inference. **LAMOL** (Sun et al., 2020) employs pseudo-replay, generating pseudo-examples of previous tasks to interleave with new task data in training, thus maintaining performance on past tasks. **VAG** (Shao et al., 2023) leverages vocabulary sparsity to selectively generate relevant outputs for each task through label generation, rather than traditional classification objective. **JOINT** learns all the classes together as a single task. It is not a continual learning setting and its accuracy is regarded as the upper-bound accuracy of CIL.

Additionally, we compare our method against several **long-context LLMs**, where all class summaries are directly included in the final in-context learning prompt (see Section 4.3) *without* using the ECL. Specifically, we evaluate using Mistral (with a 32K context window) (Jiang et al., 2023), Llama3 (128K) (Dubey et al., 2024), and Gemini (2M) (Reid et al., 2024), each trained to support these extended context lengths. We also include comparisons with LongLlama (128K) (Tworkowski et al., 2024) and LongAlpaca (16K) (Chen et al., 2024), which adapt pre-trained LLMs for handling longer contexts.

Implementation Details: We use the **Mistral-7B** model for all experiments and ablations, except for one where we evaluate our framework with different LLMs, specified accordingly. For generating summaries, we use 20 training instances per class in the summarizing prompt. [4] In the ablation with limited data, we use all available instances when there are fewer than 20. We limit the length of the summaries to a maximum of 256 tokens. To determine the optimal k value for the ECL, we use a small validation set. Despite varying values for k across different datasets, it never exceeds 3 in any dataset during our experiments, indicating that including only 3 class summaries in the final prompt is sufficient for accurate prediction, even with 150 classes as in the

[3]Note that we do not use datasets for some other NLP tasks (e.g., dialogue generation, summarization, translation, etc.) because they are not suitable for class-incremental learning. More details can be found in Section 9.

[4]We also tested with more than 20 instances per class for summaries but observed no noticeable improvement.

Dataset	#Tasks	Fine-tuning based Methods					InCA	JOINT
		Vanilla	EWC	L2P	LAMOL	VAG		
CLINC	10	51.27 $_{\pm 1.26}$	54.22 $_{\pm 1.14}$	52.53 $_{\pm 1.72}$	58.42 $_{\pm 0.84}$	76.42 $_{\pm 0.90}$	**94.40**	97.60
Banking	7	27.77 $_{\pm 2.46}$	29.10 $_{\pm 1.78}$	25.78 $_{\pm 1.21}$	42.60 $_{\pm 1.36}$	59.34 $_{\pm 1.28}$	**84.90**	92.50
DBpedia	7	39.02 $_{\pm 2.68}$	40.30 $_{\pm 2.89}$	42.84 $_{\pm 5.47}$	48.61 $_{\pm 1.82}$	65.40 $_{\pm 1.52}$	**84.20**	95.70
HWU	8	38.38 $_{\pm 4.01}$	42.72 $_{\pm 2.62}$	28.77 $_{\pm 3.18}$	44.85 $_{\pm 1.57}$	56.88 $_{\pm 1.22}$	**86.61**	90.43

Table 1: Final accuracy (%) of InCA compared with various fine-tuning based baselines. The gray column shows the results in the JOINT setting, which is not continual learning and regarded as the **upper bound**. Experiments are conducted with three different task splits to minimize the influence of any specific task split on performance.

CLINC dataset.

For the Vanilla, EWC, and JOINT systems, we perform parameter-efficient fine-tuning using LoRA adaptors (Hu et al., 2021). This approach is selected due to the large model size and PEFT's better adaptation to the task with limited data. Following (Shao et al., 2023), we employ a label generation objective instead of a classification head, as generation loss helps mitigate CF. For LAMOL and VAG, we had to use their original language models, BART for VAG and GPT-2 for LAMOL, because LAMOL's code is incompatible with Mistral, while VAG's use of an encoder-decoder architecture also results in incompatibility. For embedding the tags, we use a small SBERT paraphrase-MiniLM-L6-v2 model. All pre-trained models are obtained from the Transformers library (Wolf et al., 2020).

Experiments were conducted on a single A100 GPU with 80GB VRAM. The baseline models implemented were trained for 10 epochs per task with a batch size of 8, employing early stopping and the Adam optimizer with a learning rate of 2e-5. The LAMOL and VAG models were executed using their official configurations and hyperparameter settings. All models were maintained at 32-bit precision during training and inference, except for long-context settings, where 8-bit quantization (Dettmers et al., 2022) was employed due to the extended prompt length and VRAM requirements.

Evaluation Metric: We measure classification accuracy after all tasks/classes have been processed, referred to as **Last or Final accuracy**. All experiments are conducted three times and the accuracy is averaged, except for the zero-training LLM setups, which are deterministic.

6 Main Results

Surpassing Traditional CIL: The proposed InCA demonstrates a clear advantage over traditional CIL methods involving training, as seen in Table 1. It significantly outperforms all the baselines across all datasets. Despite employing a range of strategies such as regularization, parameter freezing, and pseudo-replay, none of the baseline methods achieved comparable performance.

As outlined in Section 1, CF occurs when updating model parameters for a new task disrupts the knowledge acquired from previously learned tasks. Our proposed system, InCA, avoids CF as it operates without any training. Although InCA's performance remains below the upper bound achieved by JOINT fine-tuning, this is mainly due to the limitations of in-context learning, which may not match the task-specific optimization of fine-tuning.

InCA vs. Long-context Setting: The limited context window of LLMs becomes a significant challenge as the number of classes increases. InCA addresses this by using an external continual learner to identify the most relevant classes and construct a precise prediction prompt. In contrast, extending the context window of the LLM to include more information is an alternative approach. To assess the effectiveness of our approach, we compared InCA against the long-context setting of LLMs, where all class summaries are passed directly into the prediction prompt.

We conducted experiments using several long-context models, including Mistral, Llama3, and Gemini 1.5 flash, both with and without the assistance of the ECL. Additionally, we tested LongLlama and LongAlpaca, which adapt standard LLMs to handle long-context tasks. Since these models are not instruction-tuned, they are not suitable for generating the tags required by InCA. Consequently, we used these models only in the long-context setting, utilizing class summaries generated by Mistral, as their own summaries may not match the quality of instruction-tuned models.

The results, shown in Table 2, reveal that the long-context models without the ECL performed significantly worse than InCA. Even with Gemini, which has a 2M token context window, performance

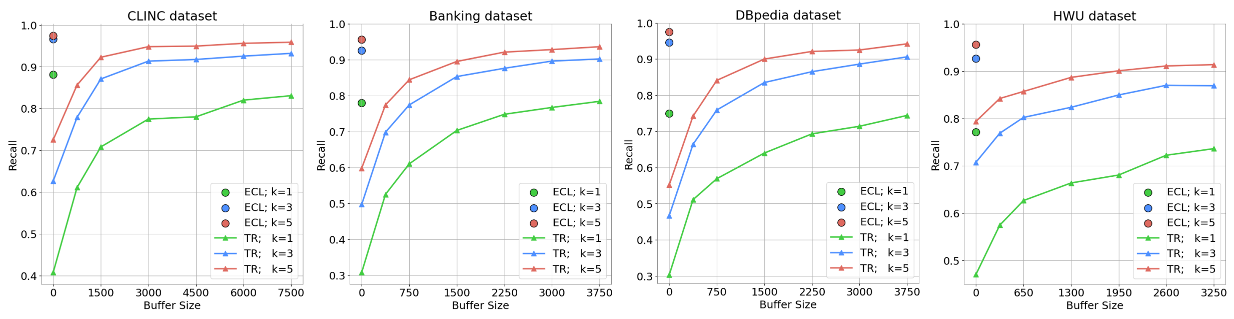

Figure 2: Comparison of recall between our ECL and the text retriever (TR) at various values of k. The ECL operates without storing any replay data (buffer size = 0), while the TR maintains a buffer of instances to retrieve the most similar ones during inference. For the TR, we evaluate performance across different buffer sizes. When the TR's buffer size is zero, we store the embeddings of class summaries for retrieval, rather than training instances.

Model	CLINC	Banking	DBpedia	HWU
Mistral-7B	94.40%	84.90%	84.20%	86.61%
Llama3-8B	95.73%	84.30%	87.60%	87.45%
Gemini 1.5 flash	95.32%	86.15%	91.63%	89.22%
	Without ECL			
Mistral-7B	86.93%	65.90%	65.30%	81.04%
Llama3-8B	83.73%	77.80%	72.70%	83.27%
Gemini 1.5 flash	93.86%	83.52%	79.64%	87.27%
LongAlpaca-7B	45.87%	33.20%	24.90%	35.97%
LongAlpaca-13B	51.20%	63.60%	59.10%	62.83%
LongLlama-3B	62.00%	52.80%	38.90%	58.46%
LongLlama-7B	84.67%	73.10%	61.00%	77.88%

Table 2: Comparison of InCA against long-context LLMs (without ECL), where all class summaries are included in the prediction prompt. For LongLlama and LongAlpaca models, class summaries are generated using Mistral, as they are not instruction-tuned.

degraded when overloaded with excessive context. This demonstrates that extending the context length is not sufficient, as overextended prompts hamper the model's ability to focus on the relevant information. In contrast, InCA ensures that only the most relevant class information (summaries) is included in the prompt, resulting in more accurate predictions and also, faster inference times due to the shorter prompts.

7 Ablation Results

Tag-based ECL vs. Text Retrieval: Given the resemblance between our approach and retrieval-augmented generation, we benchmark our tag-based classifier against a retrieval method based on text similarity. We adopt a common RAG framework, maintaining a pool of training instances for each class and using SBERT to retrieve the most similar instances during inference based on text similarity. It is important to note that this method is not applicable to CL since it involves storing original instances; we use it solely for comparison. We retrieve instances until we have obtained instances from k distinct classes, after which we measure the recall and compare these results to our ECL, which operates without a buffer (i.e., no replay data).

The results illustrated in Figure 2 demonstrate that our tag-based ECL consistently outperforms text retrieval across all scenarios, even when the text retriever has access to a substantial buffer of training instances. The same SBERT model is used for embedding both tags and input text. The superior performance of the ECL is particularly noteworthy as it does not require storing any instances, highlighting its effectiveness, especially in contexts where storing training instances is impractical.

In-context Learning Boosts Accuracy: The effectiveness of the in-context learning and the InCA approach becomes evident when comparing the model's final accuracy with the top-1 accuracy of the ECL alone. As shown in Figure 3, the final accuracy of InCA is significantly higher than the ECL's top-1 accuracy. This demonstrates the added value of in-context learning. Although the ECL alone may not be highly accurate, it effectively narrows down the relevant classes, enabling in-context learning to focus on a smaller subset and improve the overall performance.

Impact of Data Size: To assess the impact of training data size on model performance, we conducted experiments under constrained data conditions. As shown in Figure 4, we compared InCA with the JOINT fine-tuning method across varying data sizes. InCA consistently maintains performance comparable to that of the full dataset, even with significantly reduced training data (e.g., 10 examples per class). Remarkably, under these limited data conditions, InCA outperforms the JOINT fine-tuning method, often considered the upper bound. These results highlight InCA's robustness and effectiveness under limited data availability.

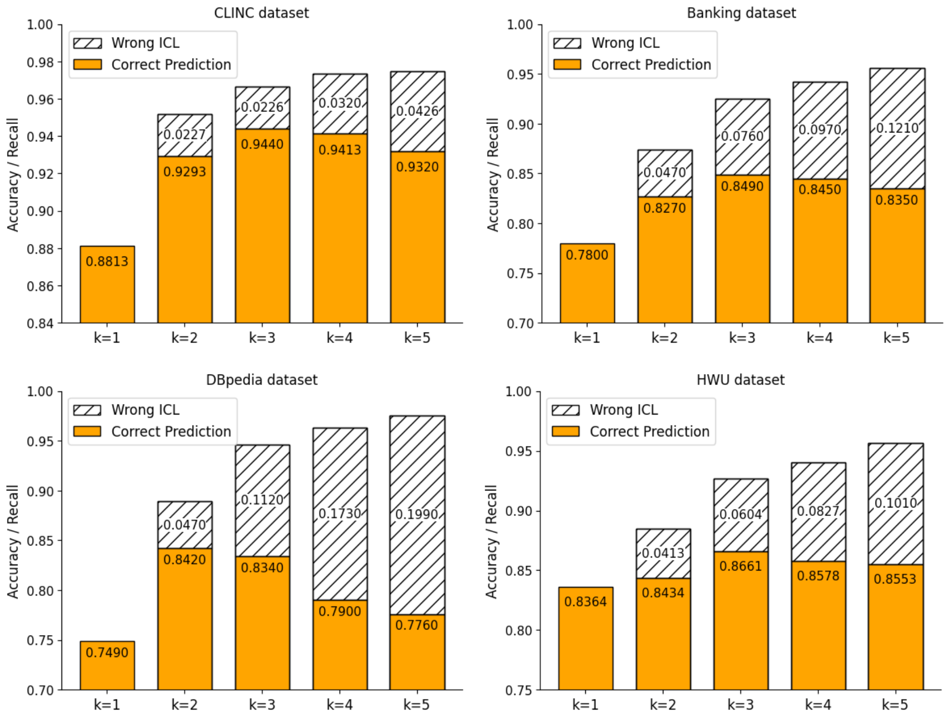

Figure 3: Accuracy of InCA and ECL recall at different k values. The solid portion of each bar represents the accuracy of the model using in-context learning (ICL) with the top k classes retrieved by the ECL. The leftmost column (k=1) represents the **accuracy of ECL alone**, where the most similar class is predicted without ICL. The dashed region indicates cases where the correct label is within the top k classes retrieved by the ECL but the model's prediction is incorrect. Therefore, the total height of each bar (solid plus dashed) represents the ECL's recall of the correct classes at that k value.

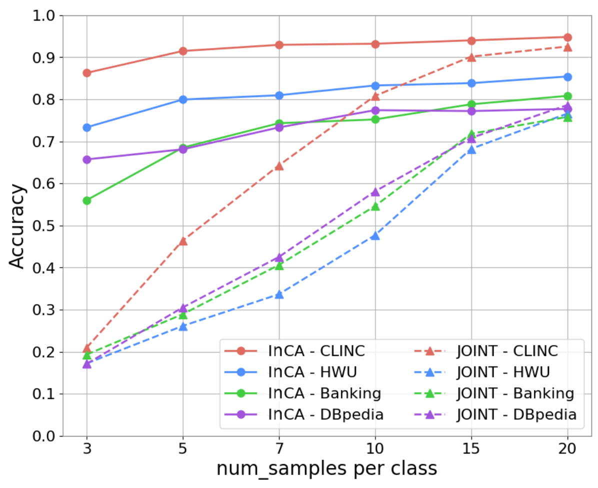

Figure 4: Performance comparison of InCA and JOINT fine-tuning across different data sizes. InCA demonstrates robust performance with limited data, particularly excelling over the fine-tuned model in data-constrained situations.

8 Conclusion

Existing continual learning (CL) research in NLP has primarily focused on fine-tuning or adapting LLMs for individual tasks, either by learning trainable prompts or adapters or updating the LLM's parameters. While these approaches can improve CL accuracy, their effectiveness remains limited due to catastrophic forgetting (CF). On the other hand, in-context learning with LLMs has proven highly effective across various NLP tasks. However, its application to CL is hindered by the limited context window of LLMs. As the number of tasks increases, the in-context prompt grows, often exceeding the token limit or leading to performance degradation due to overextended context, which may include irrelevant information. This paper proposed a novel method to address these challenges by leveraging an external continual learner. Our method is replay-free and does not fine-tune or adapt the LLM, treating it solely as a black box. Experiments show that our method markedly outperforms baselines, without suffering from CF.

9 Limitations and Future Work

One limitation of this work is that experiments are conducted exclusively on text classification datasets. This focus may limit the generalizability of our model to other types of NLP tasks (e.g., dialogue generation, summarization, translation, sentiment analysis, etc), which have different data characteristics and task requirements and are not suitable for *class-incremental learning* because they are not classification tasks with many classes that may be learned incrementally. Although sentiment analysis is often solved as a classification task, it has a fixed number of classes, i.e., positive, negative, and neutral. These other tasks are more suitable for *task-incremental learning* or *domain-incremental learning* (Ke and Liu, 2022). We believe some variations of the proposed method should apply to the other NLP tasks. Designing one general method that is suitable for multiple different NLP tasks will be an interesting future research direction.

Acknowledgments

This work was supported in part by four NSF grants (IIS-2229876, IIS-1910424, IIS-1838770, and CNS-2225427) and a research contract from KDDI Research.

References

Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. 2020. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*, pages 722–735. Springer.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. *The Twelfth International Conference on Learning Representations*.

Zhiyuan Chen and Bing Liu. 2018. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207.

Yung-Sung Chuang, Shang-Yu Su, and Yun-Nung Chen. 2020. Lifelong language knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2914–2924.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. 2000. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale, 2022. *CoRR abs/2208.07339*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Binzong Geng, Fajie Yuan, Qiancheng Xu, Ying Shen, Ruifeng Xu, and Min Yang. 2021. Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 517–523.

Claudio Greco, Barbara Plank, Raquel Fernández, and Raffaella Bernardi. 2019. Psycholinguistics meets continual learning: Measuring catastrophic forgetting in visual question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3601–3605.

Pankaj Gupta, Yatin Chaudhary, Thomas Runkler, and Hinrich Schuetze. 2020. Neural topic modeling with continual lifelong learning. In *International Conference on Machine Learning*, pages 3907–3917. PMLR.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. 2022. Demix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5557–5576.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Xu He and Herbert Jaeger. 2018. Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on Learning Representations*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization. In *Proceedings of the 2021 Conference* *of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2736–2746.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021a. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. *Proceedings of Internaional Conference and Learning Representations (ICLR-2023)*.

Zixuan Ke, Hu Xu, and Bing Liu. 2021b. Adapting bert for continual learning of a sequence of aspect sentiment classification tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4746–4755.

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. 2022. A theoretical study on solving continual learning. In *Advances in Neural Information Processing Systems*.

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, and Bing Liu. 2023. Learnability and algorithm for continual learning. In *International Conference on Machine Learning*, pages 16877–16896. PMLR.

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

Tatsuya Konishi, Mori Kurokawa, Chihiro Ono, Zixuan Ke, Gyuhak Kim, and Bing Liu. 2023. Parameter-level soft-masking for continual learning. In *International Conference on Machine Learning*, pages 17492–17505. PMLR.

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. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316.

Dingcheng Li, Zheng Chen, Eunah Cho, Jie Hao, Xiaohu Liu, Fan Xing, Chenlei Guo, and Yang Liu. 2022. Overcoming catastrophic forgetting during domain adaptation of seq2seq language generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5454.

Yuanpeng Li, Liang Zhao, Kenneth Church, and Mohamed Elhoseiny. 2019. Compositional language continual learning. In *International Conference on Learning Representations*.

Yunlong Liang, Fandong Meng, Jiaan Wang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2024. Continual learning with semi-supervised contrastive distillation for incremental neural machine translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10914–10928.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. 2022. Beyond not-forgetting: Continual learning with backward knowledge transfer. *Advances in Neural Information Processing Systems*, 35:16165–16177.

Junpeng Liu, Kaiyu Huang, Hao Yu, Jiuyi Li, Jinsong Su, and Degen Huang. 2023. Continual learning for multilingual neural machine translation via dual importance-based model division. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12011–12027.

Qingbin Liu, Xiaoyan Yu, Shizhu He, Kang Liu, and Jun Zhao. 2021a. Lifelong intent detection via multi-strategy rebalancing. *arXiv preprint arXiv:2108.04445*.

Tianlin Liu, Lyle Ungar, and João Sedoc. 2019. Continual learning for sentence representations using conceptors. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2021b. Benchmarking natural language understanding services for building conversational agents. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction: 10th International Workshop on Spoken Dialogue Systems*, pages 165–183. Springer.

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

Kun Il Park, M Park, et al. 2018. *Fundamentals of probability and stochastic processes with applications to communications*. Springer.

Chengwei Qin, Chen Chen, and Shafiq Joty. 2023. Lifelong sequence generation with dynamic module expansion and adaptation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6701–6714.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Elle: Efficient lifelong pre-training for emerging data. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2789–2810.

Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. 2020. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13588–13597.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR.

Yijia Shao, Yiduo Guo, Dongyan Zhao, and Bing Liu. 2023. Class-incremental learning based on label generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1263–1276.

Yilin Shen, Xiangyu Zeng, and Hongxia Jin. 2019. A progressive model to enable continual learning for semantic slot filling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1279–1284.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

Gido M Van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Fu-Yun Wang, Da-Wei Zhou, Liu Liu, Han-Jia Ye, Yatao Bian, De-Chuan Zhan, and Peilin Zhao. 2022a. Beef: Bi-compatible class-incremental learning via energy-based expansion and fusion. In *The Eleventh International Conference on Learning Representations*.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024a. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. 2024b. Rehearsal-free modular and compositional continual learning for language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 469–480.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. 2024c. Inscl: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 663–677.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022b. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Shipeng Yan, Jiangwei Xie, and Xuming He. 2021. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024.

Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024a. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024b. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11641–11661.

Qi Zhu, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang. 2022. Continual prompt tuning for dialog state tracking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1124–1137.

A Prompts and Examples

In this section, we detail the prompts used in various parts of our model. The provided prompts and examples are from the CLINC dataset. For different datasets, we use slightly modified versions of the same prompts based on the task at hand (e.g., intent classification, topic classification).

A.1 Summarization Prompt

Prompt

```
prompt = (
    Review the following user queries and provide a summary of the intent. Keep
    the summary generic and avoid referencing any named entities that appear in
    the queries.

    Queries: {user_queries}
    Summary:
)
```

Example

Review the following user queries and provide a summary of the intent. Keep the summary generic and avoid referencing any named entities that appear in the queries.

Queries:

- how would i apply for an american express
- show me the application for a visa card
- can you apply for idaho independent card for me
- do you have any information on how to open a new credit card
- is this where i apply for a new credit card
- is it possible to apply for a new credit card
- walk me through applying for a visa card
- what's the procedure to apply for a new credit card
- i want to put in an application for a line of a credit card how do i do that
- does bank of america give credit cards to people like me

Summary:

The intent of these queries is to apply for a new credit card. Users are looking for information on the application process, requirements, and specific credit card types. Some users are interested in applying for a specific credit card, such as an American Express or a Visa card, while others are looking for general information on how to open a new credit card.

A.2 Tag Generation Prompt

Prompt

```
prompt = (
    Generate descriptive tags for the following queries. Focus on user
    intention, relevant entities, and keywords. Extend these tags to related,
    unmentioned terms that are contextually relevant.

    Guidelines:
    Topic: Identify user intention or subject area the query pertains to.
    Entity Recognition: Focus on recognizable entities common in similar
    queries.
    Keywords: Extract specific terms or verbs that define the query's intent.
    Related Tags: Include tags that are related to user intention, even if not
    directly mentioned, to provide broader contextual understanding.

    Examples: {example_section}

    Query: {user_query}
    Tags:
)
```

Example

Generate descriptive tags for the following queries. Focus on user intention, relevant entities, and keywords. Extend these tags to related, unmentioned terms that are contextually relevant.

Guidelines:

Topic: Identify user intention or subject area the query pertains to.

Entity Recognition: Focus on recognizable entities common in similar queries.

Keywords: Extract specific terms or verbs that define the query's intent.

Related Tags: Include tags that are related to user intention, even if not directly mentioned, to provide broader contextual understanding.

Examples:

Query: "Should I wear a coat today?"
Tags: weather advice, inquiry, clothing, temperature, coat, wear

Query: "Book a table for two at a popular Italian restaurant downtown?"
Tags: dining reservation, Italian cuisine, booking, restaurant, table, request

Query: "How can I send money to a foreign bank account using the app?"
Tags: international money transfer, send money, app, foreign bank, digital transfer

Query: do i have to pay for carry-ons on delta
Tags: airline fees, carry-on, delta airlines, travel, pay, luggage

A.3 Prediction Prompt

Prompt

```
prompt = (
    Based on the given query, classify the user's intent into one of the
    following categories: {retrieved_classes}

    {class_summaries}

    Query: {user_query}
    Class:
)
```

Example

Based on the given query, classify the user's intent into one of the following categories: direct_deposit, income, payday

direct_deposit:
The users are inquiring about the process of setting up a Direct Deposit for their paychecks or bank accounts. They want to know how to arrange for their checks to deposit directly into their accounts and are looking for instructions or guidance on how to do this. Some users are specifically interested in setting up Direct Deposit at certain banks, while others are seeking general information on how Direct Deposit works.

income:
The users are inquiring about their current or past income, salary, or earnings from their job. They want to know how much money they make or earned, and sometimes they want to calculate their total income. Some users are also interested in knowing the amount they bring in annually or their compensation.

payday:
The users are inquiring about the timing of their next paycheck or payment. They want to know how often they are paid, when they can expect to be paid next, and when their next payment will be deposited. They are also interested in knowing the date or day on which they will receive their next check or be paid. Some users want to be informed about the date their most recent payment was made, while others want to plan for their next upcoming payment.

Query: get my paycheck to direct deposit
Class: direct_deposit