Prompt Augmented Generative Replay via Supervised Contrastive Learning for Lifelong Intent Detection

Vaibhav Varshney, Mayur Patidar, Rajat Kumar, Lovekesh Vig, Gautam Shroff
TCS Research, New Delhi, India
{varshney.v, patidar.mayur, k.rajat2, lovekesh.vig, gautam.shroff}@tcs.com

Abstract

Identifying all possible user intents for a dialog system at design time is challenging even for skilled domain experts. For practical applications, novel intents may have to be inferred incrementally on the fly. This typically entails repeated retraining of the intent detector on both the existing and novel intents which can be expensive and would require storage of all past data corresponding to prior intents. In this paper, the objective is to continually train an intent detector on new intents while maintaining performance on prior intents without mandating access to prior intent data. Several data replay-based approaches have been introduced to avoid catastrophic forgetting during continual learning, including exemplar and generative replay. Current generative replay approaches struggle to generate representative samples because the generation is conditioned solely on the class/task label. Motivated by the recent work around prompt-based generation via pre-trained language models (PLMs), we employ generative replay using PLMs for incremental intent detection. Unlike exemplar replay, we only store the relevant contexts per intent in memory and use these stored contexts (with the class label) as prompts for generating intent-specific utterances. We use a common model for both generation and classification to promote optimal sharing of knowledge across both tasks. To further improve generation, we employ supervised contrastive fine-tuning of the PLM. Our proposed approach achieves state-of-the-art (SOTA) for lifelong intent detection on four public datasets and even outperforms exemplar replay-based approaches. The technique also achieves SOTA on a lifelong relation extraction task, suggesting that the approach is extendable to other continual learning tasks beyond intent detection.

1 Introduction

Modern dialog systems are reliant on user intent detection wherein the system is required to classify a user utterance into one of multiple pre-defined intents. A static intent detection model is often insufficient for real-world applications because (i) User intents evolve over time (ii) Additional functionality may be added to the system and (iii) Some intents may become obsolete. Consider a conversational Information Technology (IT) helpdesk in an enterprise, where a ticket raised by an employee is classified into one of the multiple pre-defined intents (corresponding to different IT complaints) and accordingly routed by the system to the appropriate IT support staff. Each ticket comprises a textual IT problem description manually entered by an employee in natural language. In a typical IT enterprise, the helpdesk team provides support for every IT application from installation to maintenance. As applications get upgraded, and newer applications are introduced, the set of intents can change and the intent detection model must be updated.

Figure 1: Task-Incremental Lifelong Learning (A) Vs Class-Incremental Lifelong Learning (B).

In the above example, the intent detection model is required to classify both old and newly introduced intents after updation. However, privacy concerns may limit access to prior intent data, and even if prior data is available, retraining the intent detection model from scratch becomes computationally expensive due to the ever-increasing data volume. An additional problem is that the data is skewed towards prior intents. Recently, there has
been a lot of emphasis on continual learning to address these issues for computer vision (Parisi et al., 2018) and natural language processing (Biesialska et al., 2020), wherein the objective is to train a model on a sequence of novel tasks while maintaining performance on prior learned tasks, i.e., to prevent catastrophic forgetting (McCloskey and Cohen, 1989) with little or no data from old tasks.

Sun et al. (2020a); Wang et al. (2021); Madotto et al. (2021) formulate multiclass-text classification as a task-incremental learning (Task-IL) problem where the objective is to train an “incremental learner” on a sequence of tasks. As shown in Fig. 1 (A), they treat each classification task from a novel domain as a separate task introduced at each incremental step. They further assume that each task definition is complete, i.e., for each classification task they have access to all possible class labels and the set of class labels for that task remains immutable. However, in real-world applications, a task itself may evolve over time as shown in Fig. 1 (B), where, for an intent detection task in the banking domain a new set of intents are added at each incremental step based on new user needs and new functionality introduced by the service provider. Also in Task-IL, the task identity is provided at inference time, i.e., “Which task has to be performed for a given utterance?”. This restricts the job of the “task incremental learner” to choosing one of the task-specific classes as a response. This is in contrast to lifelong intent detection, where the “incremental learner” has to consider the complete set of intents that have been added so far, independent of the number of incremental steps.

To overcome catastrophic forgetting in Task-IL, LAMOL (Sun et al., 2020a) uses a PLM as the task incremental learner and uses the same learner for generative replay, i.e., for generating samples of old tasks in Task-IL. Madotto et al. (2021) propose a residual adapter (Houlsby et al., 2019) based architecture for lifelong intent detection in task-oriented dialogue systems. Unlike existing Task-IL approaches, the authors also predict task-id, i.e., which adapter to use at test time by using an entropy-based classifier. Similar to LAMOL, they also model lifelong intent detection as Task-IL where dialogues from new domains are added to the system over a period of time. However, the task-specific adapters are trained in isolation which often results in erroneous task-id predictions due to an overlap between the intents introduced over different incremental steps. Liu et al. (2021b) propose a novel strategy based on exemplar-replay (MSR) which tries to perform knowledge distillation at prediction level and feature level and uses inter-class margin loss to minimize distance between new and old class embeddings.

In the current work, we model “lifelong intent detection” as a class-incremental learning (Class-IL) (Rebuffi et al., 2017) problem where a new set of intents/classes are added over a period of time, i.e., at each incremental step. To address the issue of catastrophic forgetting during “lifelong intent detection”, we extend LAMOL to the Class-IL scenario and propose the “Prompt Augmented Generative Replay” (PAGeR) model where, similar to existing approaches (de Masson d’Autume et al., 2019; Wang et al., 2020; Qian et al., 2021; Xia et al., 2021; Wang et al., 2019; Obamuyide and Vlachos, 2019; Han et al., 2020) we do not store real samples in replay memory but instead store concept words for each intent. In successive incremental steps, we generate samples for each prior intent based on the natural language intent definition and the stored intent specific concept words via PAGeR. To further improve the quality of generative replay and overall classification performance, we perform supervised contrastive fine-tuning of PAGeR and distill knowledge from the previous version of the “lifelong intent detector” while adapting it to the new set of intents.

Our key contributions are as follows:

(i) We propose a novel incremental learning approach using prompt based classification and generative replay via a common pre-trained language model.

(ii) Our generative replay approach uses prompt based generation guided by intent specific concept words and intent definition, yielding high quality samples.

(iii) To boost the quality of generative replay, we propose an approach for supervised contrastive fine-tuning of pre-trained Language Models (PLMs).

(iv) Our approach, PAGeR, outperforms exemplar replay based approaches for lifelong intent detection on three public intent classification datasets, one text classification dataset and two public multi-domain dialog intent detection datasets.

(v) We further demonstrate that PAGeR also outperforms the state-of-the-art on a lifelong relation extraction task by a wide margin suggesting generalizability of the proposed technique across tasks.
2 Related Work

2.1 Lifelong Learning

van de Ven and Tolias (2018) proposed three problem scenarios of increasing difficulty for evaluating continual learning algorithms based on whether the task identity is known or not and if not known whether it has to be predicted at inference time or not. The three scenarios correspond to (i) Task Incremental Learning (Task-IL) (ii) Domain-incremental learning (Domain-IL) and (iii) Class-incremental learning (Class-IL) respectively. Among these three scenarios, Class-IL is the most challenging whereas Task-IL is comparatively the easiest. Different approaches have been proposed to alleviate catastrophic forgetting for continual learning including Rehearsal-based methods where a subset of labelled training data from prior tasks is stored in a memory buffer and later replayed while training the model on the current task. Chaudhry et al. (2019b) proposed Experience Replay (ER) where the data from prior tasks is interleaved with data from the current task to retain performance on old tasks. GEM (Lopez-Paz and Ranzato, 2017) ensures that at every training step the loss of each previous task, approximated by the samples in episodic memory, does not increase. whereas A-GEM (Chaudhry et al., 2019a) attempt to ensure that at every training step the average episodic memory loss over previous tasks does not increase. EWC (Kirkpatrick et al., 2017) is an example of a Regularization based approach which remembers old tasks by selectively slowing down learning on the weights important for old tasks. iCaRL (Rebuffi et al., 2017) uses prototype rehearsal along with Knowledge distillation to alleviate catastrophic forgetting in a Class-IL scenario along with a herding (Welling, 2009) based approach for prioritized exemplar selection. DER (Buzzega et al., 2020) combines rehearsal with knowledge distillation and regularization to alleviate catastrophic forgetting in all three scenarios by matching the network’s logits sampled throughout the optimization trajectory. In architecture based approaches the model is expanded for each new task with task-specific components (Rusu et al., 2016) or the same model is used for different tasks by identifying appropriate subnetworks (Wortsman et al., 2020). Shin et al. (2017) provide a Generative Replay based approach where pseudo-labelled data corresponding to the prior task is generated and interleaved with data from the current task to train a solver whereas van de Ven et al. (2020) use the same model as both a solver and generator and instead of replaying actual samples, replay latent representations.

2.2 Rehearsal-based Methods

Monaikul et al. (2021); Qian et al. (2021); Cao et al. (2021) use different variants of knowledge distillation and representative sample selection methods to alleviate catastrophic forgetting for lifelong learning of NER (Monaikul et al., 2021), NMT (Qian et al., 2021) and hate speech classification on social media (Cao et al., 2021). For lifelong language learning on stream of text examples, de Masson d’Autume et al. (2019) use sparse experience replay, local adaptation during training, and inference respectively. Wang et al. (2020) extend the work of de Masson d’Autume et al. (2019) and propose a sparse experience replay augmented meta-learning based approach to learn better initialization for local adaptation. EA-EMR (Wang et al., 2019), EMAR (Han et al., 2020) are different methods for the lifelong relation extraction task described in section 6. Xia et al. (2021) propose a novel few-shot class incremental text classification task and model text classification as textual entailment.

2.3 Generative Replay based Methods

Sun et al. (2020a) propose an approach for TASK-IL where they generate pseudo-labelled samples corresponding to old tasks for replay and use the same model for generating a label given a sample and task identifier. Sun et al. (2020b); Chuang et al. (2020) use a common classifier-generator model LAMOL with distillation (Sun et al., 2020a) for TASK-IL.

2.4 Prompting PLMs

(Radford et al., 2019) demonstrate the possibility of solving multiple NLP tasks in the zero-shot setting by expressing each of them via a task specific natural language prompt where the corresponding output is generated in an autoregressive fashion. For a given task, there was still a significant performance gap between zero-shot and supervised learning. To reduce this gap, Schick and Schütze (2021) propose an approach where they express text classification and natural language inference in a cloze-style format and fine-tune the PLM in a few-shot setting. The same approach has since been attempted for various NLP tasks (Liu et al., 2021a).
In the context of Task-IL, LAMOL expresses different NLP tasks in the format proposed by (McCann et al., 2018) and fine-tune GPT-2 on the entire task-specific data. In PAgEr, we propose three prompts, two for expressing intent detection and one for labelled data generation for Class-IL. Pseudo-labelled utterance generation is guided by intent labels along with concepts words which are stored in memory instead of actual samples. We jointly fine-tune GPT-2 on an intent detection and labelled utterance generation task. To minimize the likelihood of incorrect (Utterance, label) pairs, unlike (Chen et al., 2020; Wu et al., 2020; Khosla et al., 2020; Gunel et al., 2021), we fine-tune GPT-2 on a supervised contrastive learning objective. To further alleviate catastrophic forgetting, we also use Knowledge Distillation (Hinton et al., 2015).

3 Preliminaries

Recent work has demonstrated how downstream tasks can be performed with PLMs using prompts (Liu et al., 2021a) in a zero shot setting. Typically, a prompting function \( x' = f_{\text{prompt}}(x) \) is applied to an input \( x \) to obtain the corresponding prompt \( x' \) containing \( x \), an (intermediate) answer \( z \) and task specific discrete or continuous tokens as task descriptors. For instance, for the sentiment analysis task, \( (x' = [x]. \text{The movie is } [z]) \) is an example of a prompt where \( [x]=\text{“I like this movie.”} \) and “The movie is” are task specific discrete tokens. Given a prompt, an answer \( z \) can be generated using a PLM model \( p(z | [x]) \). The movie is; \( \varphi \). Prompts with a random or true value of \( z \) are referred to as Filled Prompts and Answered Prompts respectively. Schick and Schütze (2021) propose prompt-based fine-tuning of PLMs to minimize the performance gap between supervised and zero-shot prompt approaches on downstream tasks.

Our approach is inspired by LAMOL (Sun et al., 2020a) which fine-tunes a GPT-2 PLM over a sequence of tasks. In LAMOL, each task is represented as a prompt generated via a prompt function \( f_{\text{prompt}}(\text{Context} = \text{Context, Question, ANS, Answer, EOS}) \), where Context corresponds to the input, Question acts as a task descriptor, Answer refers to the output and ANS, EOS refers to special tokens. LAMOL was initially used for training a single multi-task NLP model without any task-specific parameters (McCann et al., 2018). For the Sentiment Analysis task (Context: “I like this restaurant.”, Question: “Is this sentence positive or negative?”, Answer: “positive”), Context represents the input sentence/paragraph, Question includes all possible class labels and Answer represents the true label. Similar to generative pre-training (Radford and Narasimhan, 2018) of a transformer based decoder only model, LAMOL performs prompt based fine-tuning of GPT-2 by maximizing \( P(\text{Answer} | \text{Context, Question, ANS}) \) for answer generation and \( P(\text{Context, Question, ANS, Answer, EOS} | \text{Task}) \) for generative replay. At inference time, LAMOL generates a class label given a Context and task-identifier, i.e., \( P(\text{Answer} | \text{Context, Question, ANS}) \). In LAMOL, generative replay is conditioned solely on task-specific tokens. Similar to LAMOL, we use GPT-2 as PLM for both lifelong intent detection and generative replay.

4 Problem Description

In this paper, we model the “lifelong intent detection” problem as a class-incremental learning problem where a set of new intents are introduced at each incremental step. Consider \( D^j = \{T_{j1}, T_{j2}, ..., T_{jm}\} \) and \( T_{jk} = \{\bigcup_{k=1}^{r}(x_{jk}^1, y_{jk}^1)\} \) where \( D^j = D^j_{\text{train}} \cup D^j_{\text{val}} \cup D^j_{\text{test}} \) represents the set of new intents and \( T_{jk} \) represents labelled data corresponding to the \( j^{th} \) intent introduced at the \( i^{th} \) incremental step respectively. \( x_{jk}^1 \) represents the \( k^{th} \) natural language utterance corresponding to the \( j^{th} \) intent introduced in the \( i^{th} \) incremental step and the corresponding intent label is represented by \( y_{jk}^1 \). Unique intent labels introduced at the \( i^{th} \) incremental step are represented by \( Y^i = \{y_{i1}^1, y_{i2}^1, ..., y_{im}^1\} \) and there will not be any overlap between the intents introduced at different incremental steps, i.e., \( Y^i \cap Y^{i'} = \emptyset \) where \( t \in \{1, 2, ..., i − 1\} \) and \( Y^i_{\text{all}} \) represent unique intent labels introduced so far where \( Y^i_{\text{all}} = Y^i \cup Y^{i−1} \cup ... \cup Y^1 \). At the \( i^{th} \) incremental step the whole labelled data corresponding to previous intents, i.e., \( D^i_{\text{all}} = D^i_{\text{train}} \cup D^{i−2}_{\text{train}} \cup ... \cup D^{1}_{\text{train}} \) is not available for training but a fixed size memory \( M \ll |D^i_{\text{all}}| \) is accessible by the incremental learner. We represent the incremental learner trained at the \( i^{th} \) incremental step by \( L^i \). And the \( L^i \) is evaluated on \( D^i_{\text{test-full}} = D^i_{\text{test}} \cup D^{i−2}_{\text{test}} \cup ... \cup D^{1}_{\text{test}} \) which includes utterances corresponding to all the intents seen so far in addition to new intents.

\( \mathcal{I} = \{N^1, N^2, ..., N^7\} \) represents the num-
ber of intents introduced over each incremental step where \( N_i \) represents number of new intents introduced at \( i^{th} \) incremental step and \( T \) represents the total number of incremental steps for a dataset \( D = D^1 \cup D^2 \cup \ldots \cup D^T \). Thus at the \( i + 1^{th} \) incremental step the objective is to train a learner \( \mathcal{I}_{L+1} \) given \( D^{train}_{i+1} \), \( M \) and evaluate it on \( D^{test\rightarrow full}_{i+1} \).

5 Proposed Approach

In PAGeR, we model “Lifelong Intent Detection” as a text generation problem, as shown in Fig. 2 (A), where the objective is to generate the correct label \( y_j^{i+1} \) for intents given a user utterance \( x_j^{i+1} \), while also utilizing generative replay to maintain performance on old intents. At the \( i + 1^{th} \) incremental step, we jointly fine-tune \( \mathcal{I}_{L+1} \) on the “Lifelong Intent Detection” \( LID \) and “Labelled Utterance Generation” \( LUG \) task given \( D^{train}_{i+1}, S^i \) which consists of labelled \( (D^{train}_{i+1}) \) and pseudo-labelled \( (S^i) \) intents respectively. Pseudo-labelled utterances corresponding to intents present in \( D^{old} \) are generated from \( \mathcal{I}_i \) based on prompts stored in a fixed size memory, \( M \) as shown in Fig. 2 (B). To minimize the likelihood of incorrect \((x_j^{i+1}, y_j^{i+1})\) pairs, we do supervised contrastive fine-tuning of \( \mathcal{I}_{L+1} \), as shown in Fig. 2 (C). To alleviate catastrophic forgetting, we use knowledge distillation (Hinton et al., 2015) by fine-tuning \( \mathcal{I}_{L+1} \) on \( S^i \) with soft targets on \( LID \) and \( LUG \) whereas we use hard targets while fine-tuning \( \mathcal{I}_{L+1} \) on novel intents in \( D^{train}_{i+1} \).

For the remainder of this section, we drop the superscript \( i + 1 \) for simplicity which indicates the \( i + 1^{th} \) incremental step. We use \((x, y)\) for \((x_j^{i+1}, y_j^{i+1})\) and refer to the \( i + 1^{th} \) step as the current incremental step.

5.1 Lifelong Intent Detection \((LID)\)

In Task-IL (van de Ven and Tolias, 2018), a task-identifier is provided at inference time, and the model performs the corresponding task. In Class-IL, no class/task information is provided a priori which makes the problem more challenging and not amenable to existing models such as LAMOL. We propose two ways of creating a prompt so that we can extend LAMOL to Class-IL.

(i) Prompt without question (PWQ): In PWQ, we convert each user utterance \( x \) in a prompt by applying a pre-defined template as mentioned in Eq 1. An Answered Prompt corresponding to PWQ is shown in Fig. 3 (A).

(ii) Prompt with incremental question (PWQ): In PWQ, we transform each user utterance \( x \) into a prompt by application of a pre-defined template as mentioned in Eq 2. where \( IncQ \) at the current incremental step includes all the intent labels \( Y^{i+1}_{all} \) seen so far. The Answered Prompt corresponding to \( PWQ \) is shown in Fig. 3 (B).

\[
f^{PWQ}_{prompt}(x) = (x, ANS, y, EOS) \quad (1)
\]

\[
f^{PWQ}_{prompt}(x) = (x, IncQ, ANS, y, EOS) \quad (2)
\]

ANS, EOS refers to special tokens used during prompt creation. We perform \( PWQ \) based fine-tuning of \( \mathcal{I}_i \) on \( D^{train}_{i+1} \cup S^i \) to arrive at the \( \mathcal{I}_{L+i+1} \) by maximizing \( \log(p(y, EOS \mid x, ANS; \varphi)) \) or by minimizing the loss function as mentioned in Eq. 3.

\[
L_{ID} = -\log(p(y, EOS \mid x, ANS; \varphi)) \quad (3)
\]

5.2 Labelled Utterance Generation \((LUG)\)

To generate high quality pseudo-labelled utterances for generative replay in subsequent incremental steps, we condition our intent specific generative model on an “intent label” and the corresponding \( Concepts \). For each intent \( I_j \), we obtain the top-K tf-idf words \( t_f - idf_j \) from the utterances corresponding to intent \( I_j \) to create a prompt. We use a maximum of \( K = 50 \) tf-idf words per intent. Words which are part of both \( x \) and \( t_f - idf_j \) are referred to as intent \( Concepts \) and \( C_j = \{c_{j1}, c_{j2}, \ldots, c_{jr}\} \) represents a set of \( Concept \) sequences one per labelled utterance present in \( I_j \). We use the template outlined in Eq. 4 to create a prompt for \( LUG \) and the corresponding Answered Prompt as shown in Fig. 3 (C). We train \( \mathcal{I}_{L+1} \) for prompt based generation by minimizing the loss function in Eq. 5.

\[
f^{LUG}_{prompt}(y, c) = (y, c, SEP, x, ANS, y, EOS) \quad (4)
\]

We use \( \mathcal{I}_i \) along with \( \mathcal{M} \) to generate the pseudo-labelled data \( S^i \) corresponding to intents present in \( D^{old}_i \) for training of \( \mathcal{I}_{L+i+1} \). Existing replay based approaches either store a few labelled utterances per intent in \( \mathcal{M} \) or generate an equal number of pseudo-labelled utterances. PAGeR, instead uses the same memory to store tf-idf concept
Figure 2: Proposed approach for Lifelong Intent Detection, where a PLM is jointly fine-tuned on (A) Intent detection, for generating the “intent label” corresponding to a user utterance (B) Generative replay, i.e., given a prompt consisting of the “intent” and a representative set of words from that intent, generate a user utterance along with the corresponding “intent label”(C) Supervised contrastive learning, i.e., given a valid utterance intent pair \((U, I)\) and another randomly chosen intent \(I_2\), predict whether \(I_1\) and \(I_2\) correspond to the same intent. For E.g., (“How do I link my new card?”;“Card Linking”) is an example of an (utterance,intent) pair and “Cash Withdrawal Charge” refers to a (possibly different) intent.

words from \(C_j\) for every intent along with \(y_j\) and generates pseudo-labelled utterances corresponding to each entry in memory.

\[
L_R = -(\log(p(x, ANS, y, EOS | y, c_{jk}, SEP; \varphi)))
\]

(5)

5.3 Supervised Contrastive Training (SCT)

To minimize the likelihood of incorrect utterance-label pairs during \(LID\) and \(LUG\), we explicitly fine-tune \(I_L + 1\) on positive and negative utterance-label pairs via contrastive loss as shown in Eq. 6. A pair \(((x, y), y')\) where \(y' \in Y_{all}\) is positive if \(y = y'\) or negative otherwise. \(p_{ul} = \text{sigmoid}(W_{c}h_{EOS} + b_{c})\), represents the probability that \(((x, y), y')\) is positive and \(h_{EOS}\) is a representation of \(EOS\) token from the PLM. \(W_{c}\) and \(b_{c}\) represents the weight and bias of a linear layer respectively.

\[
L_{SCT} = p_{ul} * \log(p_{ul}) + (1 - p_{ul}) * \log(1 - p_{ul})
\]

(6)

5.4 Knowledge Distillation (KD)

To alleviate catastrophic forgetting, we also use KD (Hinton et al., 2015) in the context of \(LID\) where instead of hard targets we use soft targets to train \(I_L + 1\) on \(S_{1}\). For intent detection, we distill the knowledge from \(I_L\) while training \(I_L + 1\) by minimizing the loss function in Eq. 7 where \(p'(y_{s+1})\)

is obtained from \(I_L\) and \(y_0 = ANS\).

\[
L_{KD} = -\frac{1}{N} \sum_{i=1}^{N} p'(y_{s+1})\log(p(y_{s+1} | x, y_0, ..., y_s; \varphi))
\]

(7)

At the current incremental step, we incrementally update \(I_L\) based on \(D_{train} + \sum\) by minimizing \(L_{total}\) as shown in Eq. 8 to get \(I_L + 1\).

\[
L_{total} = \lambda_1 * L_{ID} + \lambda_1 * L_{KD} + \lambda_2 * L_{R} + \lambda_3 * L_{SCT}
\]

(8)

where \(\lambda_1, \lambda_2, \lambda_3\) are the set of hyperparameters. At inference, we use \(I_L + 1\) to generate intent labels for all utterances in \(D_{train + full}\).

We define \textit{PAGeR-Base} as a baseline for \textit{PAGeR}, which extends LAMOL for \textit{Class-IL}. We train \textit{PAGeR-Base} on \(\lambda_1 * L_{ID} + \lambda_2 * L_{R}\) as our loss function where replay samples are generated based on the \textit{PWQ} prompt.

6 Experimental Setup

In this section, we describe the various datasets, baseline approaches, evaluation metrics and training details. We subsequently present the results and discuss our key observations. For further training details, please refer to A.2 in appendix.

6.1 Dataset Description

\textit{Lifelong Intent Detection: CLINC150} (Larson et al., 2019) is a crowdsourced multi-domain (10
domains such as utility, travel etc.) intent detection dataset. **HWU64** (Liu et al., 2019) is a crowdsourced multi-domain (21 domains such as alarm, cooking etc.) intent detection dataset. **BANKING77** (Casasantea et al., 2020) is an imbalanced, fine-grained intent detection dataset from the banking domain. **SGD** (Rastogi et al., 2020) and **MWOZ** (Budzianowski et al., 2018) are multi-domain dialogue datasets which consist of dialogues from 19 and 8 domains respectively.

**Text Classification:** **Stackoverflow** (**S20**) (Xu et al., 2017) is a multi-class classification dataset comprising of 20,000 question titles from Stackoverflow each tagged with one of 20 different tags. **Lifelong Relation Extraction:** **FewRel** (Han et al., 2018): is a relation extraction dataset where the objective is to predict a relation type between a pair of entities in a given sentence/paragraph. For all datasets, we use standard train-test split provided by respective authors (shown in Table 5). For further dataset details, please refer to A.1 in appendix.

### 6.2 Baseline Approaches

We use **Upper Bound (UB)** where we assume that data corresponding to all incremental steps is available apriori and fine-tune \( \mathcal{I} \mathcal{L} \) on all \( \mathcal{D}^t \)'s together and in **Lower Bound (LB)**, we incrementally fine-tune \( \mathcal{I} \mathcal{L} \) on \( \mathcal{D}^t \)'s with \( |\mathcal{M}| = 0 \).

#### 6.2.1 Lifelong Intent Detection

(i) **EWC** (Kirkpatrick et al., 2017): Remembers old tasks by selectively slowing down learning on the weights important for these tasks.

(ii) **A-GEM** (Chaudhry et al., 2019a): A-GEM tries to ensure that at every training step the average episodic memory loss over the previous tasks does not increase.

(iii) **ER** (Rolnick et al., 2019): Interleaving old samples with current data in training batches.

(iv) **DER** (Buzzega et al., 2020): Rehearsal with knowledge distillation and regularization.

(v) **AdapterCL** (Madotto et al., 2021): Train a task-specific residual adapters (Houlsby et al., 2019) and uses an entropy-based classifier to select which adapter to use at test time.

#### 6.2.2 Lifelong Relation Extraction

(i) **GEM** (Lopez-Paz and Ranzato, 2017): GEM ensures that at every training step the loss for each of the previous tasks, approximated by the samples in episodic memory, does not increase.

(ii) **EMR** (Parisi et al., 2019): Jointly train a model on current and old labelled samples stored in memory.

(iii) **EA-EMR** (Wang et al., 2019): They utilize an explicit alignment model to mitigate the sentence embedding distortion of the learned model when training on new data and new relations.

(iv) **EMAR** (Han et al., 2020): Every time neural models are activated to learn both new and memorized data, EMAR utilizes relation prototypes as a memory reconsolidation exercise to keep a stable understanding of old relations.

### 6.3 Evaluation Metrics

We use the following two metrics for evaluating performance on the lifelong intent detection task.

**Average Accuracy** (Chaudhry et al., 2018)

We use \( \text{Avg}^{\text{acc}} \) to evaluate the performance after \( k^t \) incremental steps where \( \text{Acc}_k^{\text{avg}} = \frac{1}{k} \sum_{i=1}^{k^t} A(\mathcal{I} \mathcal{L}_{k^t}(\mathcal{D}^t_{\text{test}})) \) and \( A(\mathcal{I} \mathcal{L}_{k^t}(\mathcal{D}^t_{\text{test}})) \) represents classification accuracy obtained by \( \mathcal{I} \mathcal{L}_{k^t} \) on \( k^t \) incremental steps on \( \mathcal{D}^t_{\text{test}} \).

**Average Forgetting (F)** (Chaudhry et al., 2018)

We use \( F^{\text{avg}} \) to measure the average drop in intent classification accuracy for intents introduced after \( k^t \) incremental steps where \( F^{\text{avg}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{k^t} \max_{1 \leq i \leq k^t} \left( A(\mathcal{I} \mathcal{L}_{k^t}(\mathcal{D}^t_{\text{test}})) - A(\mathcal{I} \mathcal{L}_{k^t}(\mathcal{D}^t_{\text{test}})) \right) \) and \( F^{\text{avg}} \in [-1, 1] \) (lower value of \( F^{\text{avg}} \) implies less forgetting).

### 7 Results and Discussion

For lifelong intent detection task, we compare \( \text{PAGeR} \) with SOTA approaches for **Class-IL** from the NLP and vision community on three imbalanced and one balanced intent detection dataset. The comparison was across three different sizes for \( \mathcal{M} \). For a given \( \mathcal{M} \), we run all baselines, including \( \text{PAGeR-Base} \) and \( \text{PAGeR} \) over four different intent permutations and report \( \text{Acc}^{\text{T}} \) and \( F^{\text{T}} \).

As shown in Table 1, \( \text{PAGeR} \) outperforms the baselines for all possible dataset-\( \mathcal{M} \) combinations. In low memory scenarios, i.e., \( \mathcal{M} = 1\% \), \( \text{PAGeR} \) outperforms exemplar replay based baselines by high margins which indicates that while it’s not possible to store all representative samples in a given memory, these may be generated based on concept words. Similar to (Buzzega et al., 2020), we also find **EWC** and **A-GEM** less effective in the **Class-IL** setting. With \( \mathcal{M} = 1\% \), \( \text{PAGeR} \) also outperforms **AdapterCL** on all datasets except **C150I**. We also compare **PAGeR** with MSR (Liu et al., 2021b) and
Table 1: Average Accuracy (↑) / Average Forgetting (↓) on four intent detection dataset, C150\(^{B1}\), C150\(^{I}\): Balanced and Imbalanced version of CLINC150, BANKING77 (B77), HWU64 and on one Text Classification (TC) dataset.

show the results in table 8. 

PAGeR also outperforms other baselines for incremental text classification on the S20 dataset. Even with \( M = 1\% \), PAGeR gets an average accuracy, \( \text{Acc}_{\text{avg}} \) of 84.3\% which is only 6.5\% below the UB. For S20, we use numeric label ids as class labels to demonstrate that PAGeR also works when labels are not informative.

### 7.1 PWQ Vs PWIQ

As shown in Fig. 4, the length of PWIQ prompt depends on the number of intent labels which increases at every incremental step and leads to large sequence length whereas the length of PWQ prompt is independent of the number of intent labels. And large sequence length leads to more running time. Hence, PWQ prompt can be extended to operate over a large number of incremental steps.

![Figure 4: PWQ Vs PWIQ](image)

**Figure 4: PWQ Vs PWIQ**

### 7.2 How good are pseudo-labelled utterances?

To alleviate catastrophic forgetting in the LID and LUG tasks, we use pseudo-labelled utterances for generative replay. To empirically assess the quality of pseudo-labelled utterances over incremental steps, we perform the following experiment: We only consider intents that have been introduced at the first incremental step. On these intents, we train an intent detection model at each incremental step where we use pseudo labelled utterances generated by \( \mathcal{I}_i \) at \( i \)-th incremental step and compare it with the model trained on real labelled-utterances (Upper Bound). As shown in Fig. 5, on the B77 dataset, the drop in average accuracy is not significant for these intents over incremental steps. This suggests that pseudo-labelled utterances generated by the model are close to original utterances, i.e., generative replay plays an important role towards alleviating catastrophic forgetting in both LID and LUG tasks.

![Figure 5: Accuracy for intents introduced during first incremental step at subsequent incremental steps](image)

**Figure 5: Accuracy for intents introduced during first incremental step at subsequent incremental steps**

### 8 Ablation Study

We conduct ablation experiments on the B77 dataset across all memory sizes to demonstrate the efficacy of PAGeR’s sub-components as shown in Table 2. Individually, all three components when applied with PAGeR-Base, show significant improvement in performance (\( \text{Acc}_{\text{avg}} \)). LUG, when applied with PAGeR-Base to generate better representative samples for replay, boosted the \( \text{Acc}_{\text{avg}} \) of PAGeR-Base by 4.7\%, 4.9\% and 5.7\% in experiments where \( M \) is set to 10\%, 5\% and 1\% respectively. Applying SCT with PAGeR-Base also shows significant improvement in experiments.
across different sizes for $\mathcal{M}$. Results of PAGeR-Base with KD only, affects the $Acc_{avg}^{T}$ of PAGeR-Base by -0.5%, -0.4% and +1.0% in experiments when $\mathcal{M}$ is 10%, 5% and 1% respectively suggesting that usage of KD by itself is not effective in alleviating catastrophic forgetting.

In row PAGeR w/o LUG, removing LUG from the combined approach highlights the importance of LUG and how the other two components, SCT and KD, are complementary to each other. This is reflected in the improved performance of PAGeR-Base by 6.8%, 7.6% and 10.4% in experiments with $\mathcal{M}$ set to 10%, 5% and 1% respectively. A similar trend is proven by rows PAGeR w/o SCT, PAGeR w/o KD, suggesting how all of these components are complementary to each other. In case of small memory size ($\mathcal{M}$=1%), all three components are required to achieve good intent detection accuracy, i.e., there is a drop of 1.5%, 3.4%, and 5.3% if we remove LUG, SCT and KD from PAGeR.

When all components are applied together with PAGeR-Base to give PAGeR, the $Acc_{avg}^{T}$ improves from 82.9% to 90.2%, 81.1% to 89.2%, and 68.1% to 80.1% with $\mathcal{M}$ set to 10%, 5% and 1% respectively.

9 Applications of PAGeR

<table>
<thead>
<tr>
<th>Approach</th>
<th>$\mathcal{M}$=10%</th>
<th>$\mathcal{M}$=5%</th>
<th>$\mathcal{M}$=1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGeR-Base (LAMOL) (A)</td>
<td>82.9</td>
<td>81.1</td>
<td>68.1</td>
</tr>
<tr>
<td>A w/o LUG only</td>
<td>87.6</td>
<td>86.0</td>
<td>73.8</td>
</tr>
<tr>
<td>A w/o SCT only</td>
<td>87.9</td>
<td>86.2</td>
<td>72.1</td>
</tr>
<tr>
<td>A w/o KD only</td>
<td>82.4</td>
<td>80.7</td>
<td>69.1</td>
</tr>
<tr>
<td>B w/o LUG</td>
<td>89.7</td>
<td>88.7</td>
<td>78.5</td>
</tr>
<tr>
<td>B w/o SCT</td>
<td>88.8</td>
<td>88.0</td>
<td>76.7</td>
</tr>
<tr>
<td>B w/o KD</td>
<td>90.0</td>
<td>87.7</td>
<td>74.8</td>
</tr>
<tr>
<td>PAGeR (B)</td>
<td>90.2</td>
<td>89.2</td>
<td>80.1</td>
</tr>
</tbody>
</table>

Table 2: Ablation on components of PAGeR on BANKING77 dataset. In PAGeR-Base (LAMOL), generative replay is based on only intent labels whereas in PAGeR-Base (LAMOL) + LUG, we use intent specific concept words along with intent labels for generative replay.

9.1 Lifelong Relation Extraction

We applied PAGeR on a lifelong relation extraction task with the same experimental setup as described in Han et al. (2020), where the objective is to identify the correct relation mentioned in a given sentence from a set of candidate relations. To utilize candidate relations in PAGeR, we use PWD instead of PWD and replace User Utterance, Incremental Question and Intent with sentence, candidate relations and correct relation respectively as outlined in Fig. 3 (B). PAGeR outperforms the state-of-the-art baseline EMAR by 13.4% in terms of $Acc_{avg}^{T}$ and is only 1.4% below the UB with $\mathcal{M}$ = 50 per incremental step, as shown in Table 3. These results suggest that PAGeR can be extended to other class incremental NLP problems.

9.2 Multi-domain Dialog

We also evaluate PAGeR in a scenario where new intents are added as part of new domain at each incremental step and we use $K$ previous utterances along with current utterance for intent detection. We use SGD, MWOZ multi-domain dialogue datasets and compare PAGeR with AdapterCL. As shown in Table 4, in this scenario, PAGeR outperforms AdapterCL.

Table 3: Average Accuracy on Lifelong relation extraction task. Except UB and LB, we have taken all baseline results from (Han et al., 2020)

<table>
<thead>
<tr>
<th>$\mathcal{M}$</th>
<th>Approach</th>
<th>FewRel</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 labelled samples per incremental step</td>
<td>UB</td>
<td>92.7</td>
</tr>
<tr>
<td></td>
<td>LB</td>
<td>86.2</td>
</tr>
<tr>
<td></td>
<td>EWC</td>
<td>30.2</td>
</tr>
<tr>
<td></td>
<td>GEM</td>
<td>59.8</td>
</tr>
<tr>
<td></td>
<td>A-GEM</td>
<td>47.5</td>
</tr>
<tr>
<td></td>
<td>EMR</td>
<td>65.1</td>
</tr>
<tr>
<td></td>
<td>EA-EMR</td>
<td>69.9</td>
</tr>
<tr>
<td></td>
<td>EMAR</td>
<td>77.9</td>
</tr>
<tr>
<td></td>
<td>PAGeR</td>
<td>91.5</td>
</tr>
</tbody>
</table>

Table 4: Average Accuracy on incremental Multi-domain intent detection task

10 Conclusion

The current paper addresses the problem of lifelong intent detection and proposes prompt augmented generative replay (PAGeR) to alleviate catastrophic forgetting. We model lifelong intent detection as a Class-IL problem and use a common PLM for intent detection and pseudo-labelled data generation which is jointly trained for both tasks yielding SOTA performance. Future work would involve demonstrating the versatility of PAGeR on a variety of different NLP tasks, and to apply it to real world Class-IL scenarios wherein the initial number of classes is large and relatively few intents are added or removed in each iteration with significant class imbalance.
References


James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu,


Qingbin Liu, Xiaoyan Yu, Shizhu He, Kang Liu, and Jun Zhao. 2021b. Lifelong intent detection via multi-strategy rebalancing.


A Appendix

A.1 Dataset Description

Lifelong Intent Detection: CLINC150 (Larson et al., 2019) is a crowdsourced multi-domain (10 domains such as utility, travel etc.) intent detection dataset comprising of 23,700 queries with 22,500 in-scope queries labelled with 150 intents and 1,200 out-of-scope queries. We only use in-scope queries, with both the balanced and imbalanced versions of the provided dataset. HWU64 (Liu et al., 2019) is a crowdsourced multi-domain (21 domains such as alarm, cooking etc.) intent detection dataset comprising of 25,716 queries labelled with 64 intents and is an imbalanced dataset. BANKING77 (Casanueva et al., 2020) is an imbalanced, fine-grained intent detection dataset from the banking domain comprising of 13,083 customer queries labelled with 77 intents. SGD (Rastogi et al., 2020) and MWOZ (Budzianowski et al., 2018) are multi-domain dialogue datasets which consist of dialogues from 19 and 8 domains respectively. Total number of intents in SGD and MWOZ are 39 and 5 respectively.

Text Classification: Stackoverflow (S20) (Xu et al., 2017) is a multi-class classification dataset comprising of 20,000 question titles from Stackoverflow each tagged with one of 20 different tags.

Lifelong Relation Extraction: FewRel (Han et al.,
2018): is a relation extraction dataset where the objective is to predict a relation type between a pair of entities in a given sentence/paragraph. It consists of 56,000 examples tagged with 80 relations introduced incrementally at each step. For all datasets, we use standard train-test split provided by respective authors (shown in table 5).

Table 5 contains details about different intent detection datasets $D$ such as CLINC150 with balanced ($C150^B$) and imbalanced version ($C150^I$), BANKING77, HWU64 (H64), and one text classification dataset, i.e., Stackoverflow (S20). It also contains details about SGD, MWOZ two multi-domain dialogue datasets which is used for intent detection task only. In these dialog datasets, the intent of an utterance can be determined on the basis of that utterance along with $K$ previous history utterances from the dialog where $K$ is hyperparameter for preprocessing during dataset creation. In our experiments, we have taken $K=0$ and $K=1$ to create the dataset for intent detection from SGD and MWOZ datasets.

Table 6 contains details about total number of incremental steps ($T$), total number of intents ($Y_{\text{fair}}^T$) in an intent detection dataset ($D$) and how these intents have been introduced over incremental steps, i.e., $I_{\text{dist}}$. And all of these datasets are available in english language and released under creative Commons licences.

Table 6: Dataset Details, where “Intents” ($Y_{\text{fair}}^T$) represents total number of intents introduced after $T$ incremental steps.

Different shuffled versions of a dataset Same set of intents are introduced over different incremental steps to create different shuffled versions of a dataset $D$, as shown in Fig. 6. We run all baselines, PAGeR and it’s variant on four different shuffled versions of $I_{\text{dist}}$ for a given $D$.

Figure 6: Different shuffled versions of BANKING77 and FewRel dataset.

A.2 Training Details
For all the experiments reported in this paper we employ the GPT-2 PLM. To ensure a fair comparison across baselines we use GPT-2 as a common backbone for all baseline models. Training for all experiments is run for a fixed number of eight epochs. During training, the batch size was constrained to lie within the range [8 – 32] and the maximum utterance length was fixed to 200 words. We used the Adam (Kingma and Ba, 2014) optimizer (with default values) for all the models and the learning rate was set to 5e-5. In all variants of PAGeR, we use top-k sampling (Holtzman et al., 2020) with $k = 20$ and $\lambda_1 = 1$, $\lambda_2 = 0.25$, $\lambda_3 = 0.25$. While synthesizing samples for contrastive learning, four negative samples were created for each positive sample. The distillation loss was optimized with a temperature value of 2. Baselines EWC, A-GEM, ER, DER are derived from the DER code repository. Code for PAGeR experiments is derived from LAMOL and “Hugging Face” (Wolf et al., 2020). All the experiments were run on Titan P100, V100 and A100 GPUs. For exemplar replay based approaches, $M$ is defined as the percentage of total training samples that can be stored for replay. $M$ is uniformly distributed across existing intents. Upon arrival of new intents, $M$ is uniformly redistributed across the total intents, both
old and new. In PAGeR, instead of real samples, we only store the corresponding concept words. For a fair comparison, we calculate the number of tokens that can be stored in the same amount of memory $M$ and we generate one pseudo-labelled utterance corresponding to each entry in $M$. We run every approach four times with shuffled versions of $D$, i.e., set of intents introduced at $T^{th}$ incremental step can be introduced at $1^{st}$ incremental step in another run. Please refer to A.3 for further details about memory management over incremental steps for different datasets.

### A.3 Memory ($M$)

To alleviate catastrophic forgetting, $M$ contains labelled utterances corresponding to old intents $Y^i_{all}$ while training the model at $i+1^{th}$ incremental step. The size of $M$ is defined in terms of the percentage of whole labelled training utterances which can be stored in it. For PAGeR, we define the memory in terms of number of tokens where the number of tokens are obtained by tokenizing the labelled utterances present in the memory at a given incremental step. Instead of storing labelled utterances in PAGeR, we only store corresponding Concepts which take less memory in terms of tokens. So in PAGeR we can store more number of Concepts in the same amount of memory as shown in Table 7.

And after $i+1^{th}$ incremental step, $M$ is equally divided among the intents seen so far, i.e., $Y^{i+1}_{all}$ and excess existing labelled utterances corresponding to old intents $Y^i_{all}$ are removed randomly.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MSR</th>
<th>PAGeR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLINC150 (C150)</td>
<td>78.00</td>
<td>92.16</td>
</tr>
<tr>
<td>HWU64</td>
<td>52.14</td>
<td>82.38</td>
</tr>
<tr>
<td>SNIPS</td>
<td>93.37</td>
<td>94.27</td>
</tr>
</tbody>
</table>

### Table 8: PAGeR experiments with fixed memory size (200)

### A.4.2 Catastrophic forgetting over incremental steps

We evaluate all baselines, PAGeR-Base and PAGeR on four shuffled versions of a dataset and report Average Accuracy along with the Standard Deviation as shown in Table 9. We observed that PAGeR gets similar results on different shuffled versions of a dataset.

It’s also evident from Fig. 7, forgetting is less in case of PAGeR as compared to other baselines as number of intents increases over incremental steps.

### Table 7: No. of Labelled Utterances/Concepts stored in $M$.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MSR</th>
<th>PAGeR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLINC150 (C150)</td>
<td>1500</td>
<td>750</td>
</tr>
<tr>
<td>HWU64</td>
<td>2964</td>
<td>1751</td>
</tr>
<tr>
<td>SNIPS</td>
<td>516</td>
<td>516</td>
</tr>
</tbody>
</table>

### A.4 Results

#### A.4.1 MSR Vs PAGeR

Liu et al. (2021b) (MSR) shows the experiments in lifelong intent detection with a fixed memory of 200 only. We performed same experiment with PAGeR and compiled the average accuracy at the last incremental step in table 8.
Table 9: Average Accuracy (↑) with standard deviation on different datasets.

<table>
<thead>
<tr>
<th>M</th>
<th>Approach</th>
<th>C150↑</th>
<th>C150↑</th>
<th>B77</th>
<th>HWU64</th>
<th>S20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UB</td>
<td>93.7</td>
<td>94.6</td>
<td>97.9</td>
<td>90.3</td>
<td>90.8</td>
</tr>
<tr>
<td></td>
<td>LB</td>
<td>36.9±3.9</td>
<td>30.1±1.4</td>
<td>23.3±2.2</td>
<td>24.4±2.9</td>
<td>21.6±2.9</td>
</tr>
<tr>
<td></td>
<td>EWC</td>
<td>36.9±4.9</td>
<td>31.6±4.0</td>
<td>21.1±2.0</td>
<td>23.8±2.4</td>
<td>20.8±1.2</td>
</tr>
<tr>
<td></td>
<td>AdapterCL</td>
<td>88.1±0.06</td>
<td>79.9±0.32</td>
<td>77.7±0.97</td>
<td>77.6±0.15</td>
<td>82.5±0.55</td>
</tr>
<tr>
<td>10%</td>
<td>A-GEM</td>
<td>30.8±10.7</td>
<td>13.6±5.8</td>
<td>17.3±5.3</td>
<td>18.8±2.9</td>
<td>67.6±3.1</td>
</tr>
<tr>
<td></td>
<td>ER</td>
<td>88.9±0.8</td>
<td>81.7±0.4</td>
<td>84.7±1.6</td>
<td>78.3±1.4</td>
<td>85.2±0.6</td>
</tr>
<tr>
<td></td>
<td>DER</td>
<td>90.8±0.9</td>
<td>83.9±0.7</td>
<td>80.7±3.7</td>
<td>81.3±2.1</td>
<td>85.6±1.4</td>
</tr>
<tr>
<td></td>
<td>PAGeR-Base (LAMOL)</td>
<td>92.7±2.9</td>
<td>89.7±2.14</td>
<td>82.9±2.9</td>
<td>84.0±1.5</td>
<td>72.5±3.5</td>
</tr>
<tr>
<td></td>
<td>PAGeR</td>
<td>94.9±2.22</td>
<td>93.4±0.37</td>
<td>90.2±0.26</td>
<td>89.9±0.67</td>
<td>88.7±3.3</td>
</tr>
<tr>
<td></td>
<td>A-GEM</td>
<td>30.8±9.1</td>
<td>13.4±1.8</td>
<td>19.2±2.9</td>
<td>18.4±3.1</td>
<td>68.2±4.4</td>
</tr>
<tr>
<td></td>
<td>ER</td>
<td>85.2±1.7</td>
<td>71.1±2.1</td>
<td>76.5±1.8</td>
<td>72.0±1.2</td>
<td>82.6±0.6</td>
</tr>
<tr>
<td></td>
<td>DER</td>
<td>88.1±0.8</td>
<td>78.1±3.4</td>
<td>73.9±2.7</td>
<td>76.5±1.2</td>
<td>83.9±0.9</td>
</tr>
<tr>
<td></td>
<td>PAGeR-Base (LAMOL)</td>
<td>92.0±0.27</td>
<td>87.5±1.45</td>
<td>81.1±3.8</td>
<td>82.2±2.3</td>
<td>71.3±2.2</td>
</tr>
<tr>
<td></td>
<td>PAGeR</td>
<td>94.4±0.35</td>
<td>92.9±0.29</td>
<td>89.2±0.48</td>
<td>88.0±0.21</td>
<td>88.0±0.6</td>
</tr>
<tr>
<td>5%</td>
<td>A-GEM</td>
<td>28.7±10.5</td>
<td>13.9±1.0</td>
<td>25.5±15.0</td>
<td>17.9±14.7</td>
<td>62.4±6.4</td>
</tr>
<tr>
<td></td>
<td>ER</td>
<td>64.8±3.2</td>
<td>41.3±1.8</td>
<td>51.6±3.7</td>
<td>46.2±0.8</td>
<td>76.3±0.8</td>
</tr>
<tr>
<td></td>
<td>DER</td>
<td>73.0±1.8</td>
<td>45.9±2.7</td>
<td>53.0±2.7</td>
<td>49.4±1.5</td>
<td>77.4±1.3</td>
</tr>
<tr>
<td></td>
<td>PAGeR-Base (LAMOL)</td>
<td>86.9±0.34</td>
<td>63.6±2.37</td>
<td>68.1±4.1</td>
<td>70.5±2.8</td>
<td>68.3±2.1</td>
</tr>
<tr>
<td></td>
<td>PAGeR</td>
<td>92.4±0.71</td>
<td>76.4±1.53</td>
<td>80.1±1.46</td>
<td>79.2±0.84</td>
<td>84.3±1.0</td>
</tr>
</tbody>
</table>

Figure 7: Average accuracy over incremental steps for different datasets. Here, average accuracy is shown on one shuffled version of each dataset.