Generative Replay Inspired by Hippocampal Memory Indexing for Continual Language Learning

Aru Maekawa¹, Hidetaka Kamigaito^{1,2}, Kotaro Funakoshi¹, Manabu Okumura¹

¹Tokyo Institute of Technology, ²Nara Institute of Science and Technology

{maekawa, kamigaito, funakoshi, oku}@lr.pi.titech.ac.jp kamigaito.h@is.naist.jp

Abstract

Continual learning aims to accumulate knowledge to solve new tasks without catastrophic forgetting for previously learned tasks. Research on continual learning has led to the development of generative replay, which prevents catastrophic forgetting by generating pseudosamples for previous tasks and learning them together with new tasks. Inspired by the biological brain, we propose the hippocampal memory indexing to enhance the generative replay by controlling sample generation using compressed features of previous training samples. It enables the generation of a specific training sample from previous tasks, thus improving the balance and quality of generated replay samples. Experimental results indicate that our method effectively controls the sample generation and consistently outperforms the performance of current generative replay methods.¹

1 Introduction

Humans and intelligent animals continually acquire new knowledge and skills throughout their lifetime. This ability, called continual learning (CL) or lifelong learning, is a fundamental requirement for human-like general intelligence (Parisi et al., 2019). CL is also crucial for practical applications, as new data and tasks to train models appear every day in the real world. It is especially important for natural language processing (NLP) systems, in which vocabulary and language usage change over time. However, most neural network based models are trained with a static dataset. When learning different tasks sequentially, performance on the previously learned tasks tends to significantly degrade, referred to as catastrophic forgetting (McCloskey and Cohen, 1989). Learning new tasks without catastrophic forgetting has been a long-standing challenge in machine learning and neural networks.

¹The source code is available at https://github. com/arumaekawa/GR-HMI. Replay is an approach to alleviate catastrophic forgetting by retraining with previous tasks' data when training a new task. Although replay-based methods are effective in most CL scenarios, it is necessary to retain training data for all previous tasks, which may cause problems with storage requirements and data privacy. Therefore, generative replay was developed (Shin et al., 2017), which uses pseudo-samples generated from generation models instead of real samples. In the NLP domain, LAMOL (Sun et al., 2020a) was proposed as a generative replay framework, where a single language model simultaneously learns to solve tasks and to generate pseudo-samples (Fig. 1a).

Although the generative replay does not require any previous task's data, it typically underperforms actual sample replay because of the balance and quality problems in sample generation. Regarding the balance problem, generation models tend to generate a large number of samples for recently learned tasks, that results in forgetting older tasks. This is due to the difficulty in controlling sample generation and catastrophic forgetting occurring in generation models. Regarding the quality problem, generative replay methods commonly assume to generate even unseen samples which is not included in the past training datasets. However, since generating such unseen samples is more difficult than generating previously trained samples, it may cause the degradation of the quality of replay samples. To prevent the catastrophic forgetting, it is sufficient to generate only previously learned samples.

To address these issues, we refer to the memory retrieval mechanism in the biological brain that achieves CL. According to the hippocampal indexing theory (Teyler and Rudy, 2007), the hippocampus encodes memory engrams for new neocortical activity patterns and uses them as memory indexes to recall past experiences. Inspired by this hippocampal mechanism, we propose hippocam-



Figure 1: (a) is the overview of the LAMOL framework. The top is the learning QA to solve tasks and the bottom is the learning LM to generate pseudo-samples. (b) is the proposed HMI implemented on LAMOL. We introduce a hippocampus module illustrated on the left.

pal memory indexing (HMI) for improving generative reply. To remember training samples with a small data usage, we introduce a hippocampus module that encodes training samples into compressed memory engrams using BERT (Devlin et al., 2019) and product quantization (PQ) (Jégou et al., 2011), and stores them to generate conditioned samples during the replay step (Fig. 1b). This method makes it possible to generate specific training samples from previously learned tasks.

We evaluated HMI on two different CL scenarios using the original LAMOL as a baseline. The first scenario is a sequence of different types of tasks, for which we used five natural language understanding (NLU) tasks from DecaNLP (McCann et al., 2018). The other scenario is a sequence of different domains in the same task, for which we used five text classification datasets and single-pass setting, which is considered as an ideal scenario for CL. The results indicate that HMI consistently outperforms LAMOL and improves robustness to training task order and amount of replay samples. We also investigated the balance of previously learned tasks in generated samples and found that HMI enables the generation of even old task samples, which indicates the controllability of sample generation with HMI. Furthermore, we explored the potential of further improvement of HMI with different sample selection strategies for replay.

2 Related Work

CL, which involves learning from a stream of tasks without catastrophic forgetting, is a long-standing issue in machine learning. In the NLP, CL has been studied for diverse tasks, for example, word and sentence representations (Xu et al., 2018; Liu et al., 2019), sentiment analysis (Chen et al., 2015; Xia et al., 2017), composition language learning (Li et al., 2020b), relation learning (Han et al., 2020), dialogue systems (Lee, 2017; Madotto et al., 2021), text classification, and question-answering (QA) (de Masson d'Autume et al., 2019; Wang et al., 2020).

Regularization-based methods aim to constrain changes in model parameters important for previous tasks. Various methods have been proposed to estimate the importance of each parameter. For example, elastic weight consolidation (EWC) (Kirkpatrick et al., 2017) uses the Fisher information matrix. Synaptic intelligence (SI) (Zenke et al., 2017) estimates importance from the contribution to loss changes. Memory-aware synapses (MAS) (Aljundi et al., 2018) computes the sensitivity of parameters on the basis of the gradient of model outputs.

Architecture-based methods dynamically change the network structure to assign model parameters for each task. Progressive neural networks (PNN) (Rusu et al., 2016) freeze the current parameters and add a new column of the network when training a new task. Instead of extending the network, PackNet (Mallya and Lazebnik, 2018) applies network pruning using dynamic filters to separate the neurons used for each task.

Replay-based methods mitigate catastrophic forgetting by retraining for previous tasks when training for a new one. MbPA++ (de Masson d'Autume et al., 2019) introduces an episodic memory that stores real samples of previous tasks to use for experience replay and local adaptation. Meta-MbPA (Wang et al., 2020) applies a meta-learning algorithm to improve MbPA++. To enhance the replaybased methods with a limited amount of samples, Wang et al. (2020) and Huang et al. (2021) also investigated effective selection strategies other than random sampling. Instead of keeping real samples for replay, Shin et al. (2017) proposed generative replay, which trains a model to generate pseudosamples. Sun et al. (2020a) proposed LAMOL as a generative replay method for NLP tasks. LAMOL uses GPT-2 (Radford et al., 2019) to simultaneously learn a variety of NLP tasks and pseudosample generation. L2KD (Chuang et al., 2020) and DnR (Sun et al., 2020b) use knowledge distillation to extend LAMOL. MFK-LAMOL (Choi and Kang, 2021) makes replay more efficient by using more forgotten pseudo-samples in generative replay. Rational-LAMOL (Kanwatchara et al., 2021) uses critical freezing guided by supervised or unsupervised rationale. RVAE-LAMOL (Wang et al., 2022) enhances LAMOL by mapping different tasks into a limited unified feature space.

Current generative replay methods have problems on the balance and quality of generated samples. To address these issues, we propose a samplegeneration control with the HMI method, inspired by the biological brain. In contrast to the previous work, our approach prevents low quality samples by using the assumption that a model generates only previously learned samples. HMI also achieves balanced sample generation by strong sample-level conditioning rather than task-level conditioning. Although our HMI can be applied to most of the existing generative replay methods, similar to other recent work, we build HMI upon LAMOL, which is a simple generative replay baseline for CL in NLP and whose implementation code is available.

3 LAMOL: Language Modeling for Lifelong Language Learning

Before describing HMI, we briefly explain LAMOL (Sun et al., 2020a), on which we propose our HMI, in this section.

LAMOL is a generative replay framework using a single GPT-2 to solve different types of NLP tasks and generate pseudo-samples. In LAMOL, all training samples are fed into GPT-2 as a sequence of *context*, *question*, and *answer*. As illustrated at the top of Figure 1a, GPT-2 learns each task in a QA manner, predicting the answer part on the basis of the given context and question. As well as training QA, GPT-2 learns language modeling (LM) to generate the whole sequence of the context, question, and answer, as illustrated at the bottom of Figure 1a. During the training step, the parameters of GPT-2, $\theta_{\text{GPT-2}}$, are optimized to minimize the QA loss \mathcal{L}_{QA} and the LM loss \mathcal{L}_{LM} together as $\mathcal{L} = \mathcal{L}_{\text{QA}} + \lambda \mathcal{L}_{\text{LM}}$, where λ is a hyperparameter.

When training for a new task, LAMOL generates pseudo-samples for previous tasks to use for replay. Assume a stream of tasks $\{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_T\}$ to train a model with LAMOL, where the number of tasks T may be unknown. Before training a new task \mathcal{T}_i (i > 1), GPT-2 generates pseudo-samples by top-ksampling from the first token [GEN]. The number of pseudo-samples is $\gamma |\mathcal{T}_i|$, where γ is the sampling ratio and $|\mathcal{T}_i|$ is the number of training samples in \mathcal{T}_i . Defective samples, which do not have a unique [ANS] token that indicates the start position of the answer, are discarded, and the others are mixed with \mathcal{T}_i to alleviate forgetting for $\mathcal{T}_{< i}$ in training.

When using the same [GEN] for all tasks, the ratio for old tasks in the generated samples decreases exponentially in theory (Sun et al., 2020a). Therefore, Sun et al. (2020a) proposed to replace [GEN] with a task-specific token [TASK] (e.g., "___sst__") to control GPT-2 to generate pseudo-samples belonging to the specific task. In the beginning of training for $\mathcal{T}_i, \frac{\gamma}{i-1} |\mathcal{T}_i|$ pseudo-samples for each previous task, $\mathcal{T}_1, \ldots, \mathcal{T}_{i-1}$, are generated using the corresponding task-specific token.

4 Hippocampal Memory Indexing (HMI)

In this section, we introduce our Hippocampal Memory Indexing (HMI) that can suppress the problems of unbalanced and low-quality generation in the replay by accessing compressed features for previous training samples.

4.1 Overview

HMI is implemented as a module that works on LAMOL. Figure 1b shows the overview of our HMI on LAMOL. The training process of HMI on LAMOL is as follows.

1. The **hippocampus module** encodes each training sample into a feature vector representation with a BERT encoder (§ 4.2.1).

2. Product Quantization (PQ) compresses the encoded feature as a **memory engram** in a **hip-pocampal memory** for future replay (§ 4.2.2). To prevent increasing the size of the hippocampal memory, we apply memory pruning at the end

of training for each task (§ 4.2.3).

3. The **memory engrams** are used to condition the generation by GPT-2 to generate the corresponding samples (§ 4.3). This allows us to control the task balance in replay samples and generate only previously learned samples of higher quality.

We describe the hippocampus module, which encodes and stores memory engrams for each training sample, in § 4.2 and explain generation control with HMI in § 4.3. Finally, we explain the pre-training for HMI in § 4.4.

4.2 Hippocampus Module

The hippocampus module is a memory component that stores memory engrams, compressed representations of training samples in previously learned tasks. During replay, the module retrieves stored memory engrams and provides them for GPT-2 to generate the corresponding previous samples. In this section, we describe the encoding of a memory engram from each training sample and the pruning of the hippocampal memory in the module.

4.2.1 Memory Engram Encoding

A memory engram is created from the context and question part of each training sample. Each sequence consisting of the context and question is first encoded to a fixed-sized vector representation with the BERT encoder. Similar to BERT, we use a special classification token [CLS] in the beginning of an input sequence to obtain a sample-level embedding $h_{[CLS]} \in \mathbb{R}^H$, where *H* is the embedding dimension in BERT. It is further converted to a *d*-dimensional feature vector $h = W_E h_{[CLS]} \in \mathbb{R}^d$, where $W_E \in \mathbb{R}^{d \times H}$ is a weight matrix. Note that, during CL, the encoder parameters $\{\phi_{\text{BERT}}, W_E\}$ are frozen to prevent the features from drifting, which degrades the controllability of HMI.

4.2.2 Feature Quantization

A feature vector $h \in \mathbb{R}^d$ is then quantized to reduce the data size of the hippocampal memory. In the beginning of training a new task, we train a quantization model on encoded feature vectors for all training samples in the new task and all feature vectors currently stored in the hippocampal memory. We use PQ (Jégou et al., 2011), which has a lower reconstruction error than a simple quantization model that uses only k-means. PQ first divides a d-dimensional h into S sub-vectors, each of dimension d/S. It then creates a codebook by computing N centroids for each S partitions with the



Figure 2: Feature-vector injection to GPT-2. (a) Self-Attention: $h_{\text{Mem},l}$ is fed into the *l*-th self-attention layer and used as an additional key and value. (b) Embedding: h_{Emb} is added to word and position embeddings.

k-means algorithm. Using this codebook, each h is quantized to *S* integer indices $c \in \{0, ..., 255\}^S$ and stored in the hippocampal memory as a memory engram.

4.2.3 Memory Pruning

It is inefficient to keep memory engrams of all training samples in each previously learned task, which may cause a scalability problem with the data size for an increased number of tasks. Thus, we set the maximum number of stored memory engrams M_{max} . When learning the *i*-th task \mathcal{T}_i , the memory engrams of all training samples in \mathcal{T}_i are first appended to the hippocampus module as the *i*-th task hippocampal memory M_i . After updating the codebook of the PQ model, every hippocampal memory $\{M_1, M_2, \ldots, M_i\}$ is reduced to satisfy $|M_j| \leq \frac{M_{\text{max}}}{i}$ for $1 \leq j \leq i$ by randomly selecting memory engrams to keep from M_j . Therefore, we can keep the total number of memory engrams at most M_{max} regardless of the number of tasks.

4.3 Sample Generation with Hippocampal Memory Indexing

To condition the sample generation, we feed the memory engram of each training sample into GPT-2. We use two schemes, each of which is based on embedding and self-attention layers (Li et al., 2020a; Fang et al., 2021), as described as follows and illustrated in Figure 2.

Embedding Layer In GPT-2, the embedding representation of the *t*-th token in an input sequence is the sum of the word embedding $\boldsymbol{h}_{\text{WE}}^{(t)} \in \mathbb{R}^{H}$ and position embedding $\boldsymbol{h}_{\text{PE}}^{(t)} \in \mathbb{R}^{H}$, where *H* is the

embedding dimension in GPT-2. The feature vector \boldsymbol{h} is added to it, so the new embedding vector is $\boldsymbol{h}_{\text{Emb}}^{(t)} = \boldsymbol{h}_{\text{WE}}^{(t)} + \boldsymbol{h}_{\text{PE}}^{(t)} + \boldsymbol{W}_{\text{D}}\boldsymbol{h}$, where $\boldsymbol{W}_{\text{D}} \in \mathbb{R}^{H \times d}$ is a weight matrix.

Self-Attention Layer The feature vector h is used as an additional key and value in each self-attention layer. It is first projected into an *LH*-dimensional vector with a weight matrix $W_{\rm M} \in \mathbb{R}^{LH \times d}$. It is then divided into *L* vectors $[h_{{\rm Mem},1}, \ldots, h_{{\rm Mem},L}]$ and converted to a key and value in each self-attention layer.

We apply the feature-vector injection to the training, replay, and inference steps. During the training step, a quantized memory engram c, encoded from a training sample in the hippocampus module, is decoded back into a feature vector h' and given to GPT-2 to condition the generation for both the QA and LM. It reduces the effect of quantization errors between training and generation to use h' rather than the BERT encoder outputting h directly. During the replay step, when training for the *i*-th task \mathcal{T}_i (i > 1), we randomly select $\frac{\gamma}{i-1} |\mathcal{T}_i|$ memory engrams from the hippocampal memory for each previous task, M_1, \ldots, M_{i-1} , and provide them for GPT-2 to generate the corresponding past training samples by greedy decoding.

In inference, GPT-2 generates the answer part on the basis of the given context and question with injected h encoded from the context and question of a test sample.

4.4 Pre-training

The memory engrams encoded in the hippocampus module should contain enough information to reconstruct the corresponding samples with GPT-2. However, a naive [CLS] token embedding of BERT and a linear projection with randomly initialized W_E are considered insufficient. In addition, the GPT-2 side uses the connections with W_D and W_M , which are initialized randomly, so it may make CL unstable, especially for initial tasks.

To address these issues, we introduce a pretraining of the BERT encoder and GPT-2 as an autoencoder (AE) using an unlabeled text corpus. In the pre-training, the BERT encoder learns to encode each input text into a single feature vector, and GPT-2 learns to reconstruct the original input from the feature vector. Note that we do not apply the quantization model to the pre-training. This is because it is not necessary to store feature vectors, since we do not consider the replay of the pre-training data, and this also allows the encoder parameters $\{\phi_{\text{BERT}}, W_{\text{E}}\}\$ to be optimized through gradient descent. With this pre-training, the model encodes well-informed memory engrams and achieves their conditional generation with GPT-2 from the beginning of CL.

Along with the AE, GPT-2 simultaneously learns the LM in this pre-training. This is to prevent GPT-2 from losing the knowledge of the LM that GPT-2 originally has by training the AE. Therefore, we optimize the sum of the AE loss and the LM loss $\mathcal{L} = \mathcal{L}_{AE} + \mathcal{L}_{LM}$, and obtain the initial model parameters { $\phi_{BERT}, W_E, \theta_{GPT-2}, W_D, W_M$ } for CL. The effect of the pre-training are given in Appendix A.

5 Experiments

5.1 General Settings

Tasks, Datasets, and Metrics To evaluate the effectiveness of HMI for CL on different types of tasks, we first evaluated it on a scenario of selecting five NLU tasks from decaNLP (McCann et al., 2018), following the settings of Sun et al. (2020a): QA (SQuAD), semantic parsing (WikiSQL), sentiment analysis (SST), semantic role labeling (QA-SRL), and goal-oriented dialogue (WOZ).

To evaluate it on another CL scenario, we used five text classification tasks from diverse domains: news classification (AGNews), sentiment analysis (Yelp, Amazon), Wikipedia article classification (DBPedia), and QA categorization (Yahoo). We followed the settings of de Masson d'Autume et al. (2019) to use the balanced version datasets² and applied the single-pass setting, where a model trains with each dataset for only one epoch.

Each dataset has the corresponding evaluation metric ranging from 0 to 100%. More details of tasks and datasets are given in Appendix B.1.

Compared Methods We compared the following methods:

- LAMOL This is the baseline generative replay method, without HMI. GPT-2 generates pseudosamples by top-k sampling with k = 20 given only first tokens [GEN]/[TASK].

- **HMI-LAMOL** Our HMI implemented on LAMOL. We evaluated the effectiveness of HMI by comparing it to LAMOL.

 $^{^{2}}$ We used the random sampled subsets released by Sun et al. (2020a).

- **Real Samples** Replay samples in HMI-LAMOL are replaced with actual training samples corresponding to the given memory engrams. This is considered as the upper bound of HMI-LAMOL in terms of the quality of generated samples.

- **Multitask** All tasks are simultaneously trained with GPT-2. Note that GPT-2 optimizes only the QA loss. Since future tasks are not accessible in CL, this is often regarded as the upper bound.

Implementation We implemented the above methods on the basis of the official implementation of LAMOL.³ We also re-implemented LAMOL to use [GEN]/[TASK] for learning the QA as well as learning the LM to make training fast and improve performance by unifying input format. We used the smallest pre-trained GPT-2⁴ as the language model for all methods and the smallest pre-trained BERT⁵ for the encoder of the hippocampus module in HMI-LAMOL.

Hyper-parameters For all methods, we followed the settings in Sun et al. (2020a): We set $\lambda = 0.25$ for the weight of the LM loss and applied greedy decoding during inference. We trained the models for nine epochs for each NLU task and for only one epoch for each text classification task. For HMI-LAMOL, we set the maximum size of the hippocampal memory M_{max} to 10,000. We also set the dimension d of the feature vector h to 768, the same as the embedding dimension in BERT and GPT-2. We set the number of partitions S = 16 and the number of centroids N = 256 for the feature quantization. All the results in our experiments were the average over two runs (seed = 42, 43). More details of the experimental settings are given in Appendix C.

For the pre-training in HMI-LAMOL (§ 4.4), we used 1,070,272 text lines from the Wiki-40B (Guo et al., 2020) test set as training data. We trained the model with them for three epochs to obtain the initial model parameters for CL.

5.2 Evaluations

5.2.1 Five Different NLU Tasks

Settings We trained each model on the five NLU tasks in either descending/ascending order in accordance with the number of training samples.

Replay sample ratio (γ)	0.01	0.05	0.2
Large to small: (SQuAD \rightarrow WikiSQL \rightarrow	$SST \rightarrow q$	2A-SRL -	$\rightarrow WOZ)$
LAMOL _{GEN} (Sun et al., 2020a)*	-	69.6	73.1
LAMOL _{TASK} (Sun et al., 2020a)*	-	71.5	74.3
LAMOL _{GEN}	69.1	74.1	75.5
LAMOL _{TASK}	67.9	74.4	76.2
HMI-LAMOL _{GEN}	72.7	75.3	76.4
HMI-LAMOL _{TASK}	72.6	75.2	76.6
HMI-LAMOL _{GEN} + Real samples	73.9	76.4	78.0
HMI-LAMOL _{TASK} + Real samples	73.7	76.4	77.6
Small to large: (WOZ \rightarrow QA-SRL \rightarrow SS	$T \rightarrow Wikk$	$SQL \rightarrow S$	SQuAD)
LAMOL _{GEN} (Sun et al., 2020a)*	-	63.2	73.0
LAMOL _{TASK} (Sun et al., 2020a)*	-	75.3	76.9
LAMOL _{GEN}	57.7	59.3	72.9
LAMOL _{TASK}	58.2	60.5	76.4
HMI-LAMOL _{GEN}	67.1	76.5	77.3
HMI-LAMOL _{TASK}	70.8	75.5	77.5
HMI-LAMOL _{GEN} + Real samples	75.5	78.0	78.9
HMI-LAMOL _{TASK} + Real samples	75.0	78.0	79.2
Multitask		77.2	

Table 1: Experimental results on the five NLU tasks in two different orders. * indicates the reported score by Sun et al. (2020a).

We evaluated it with and without task-specific tokens, and with three different replay sample ratios $\gamma \in \{0.01, 0.05, 0.2\}$. We obtained the performance of CL as the average score on the five tasks at the end of training streams.

Results Table 1 shows the results. The results indicate that HMI-LAMOL outperformed LAMOL in all cases, that is, in both two task orders and all γ . HMI-LAMOL with the best resulting setting, in ascending order and $\gamma = 0.2$, even beat multitask. HMI-LAMOL also reduced the performance gap between the two task orders. Although the performance of LAMOL degraded when the value of γ decreased, HMI-LAMOL mitigated the performance degradation of LAMOL. As expected, even when using the same [GEN] token, HMI-LAMOL typically performed as well as using task-specific tokens because it includes their role as well. However, there is also a performance gap from replacing generated samples with real samples. This indicates that HMI-LAMOL could be further improved by developing a better model for conditional generation.

³https://github.com/jojotenya/LAMOL

⁴https://huggingface.co/gpt2

⁵https://huggingface.co/

bert-base-uncased

γ	Methods	(i)	(ii)	(iii)	(iv)	Avg.	Std.
	LAMOLTASK	42.9	49.6	61.7	62.4	54.2	8.3
0.01	HMI-LAMOL _{TASK} + Real samples	68.2 70.2	68.9 71.7	67.6 70.1	69.4 71.0	68.5 70.8	0.7 0.7
0.05	LAMOLTASK	61.2	66.2	63.8	63.4	63.7	1.8
0.05	HMI-LAMOL _{TASK} + Real samples	70.9 72.7	71.7 73.2	71.1 73.3	71.1 73.1	71.2 73.1	0.3 0.2
	LAMOLTASK	71.0	71.9	68.0	71.4	70.6	1.5
0.2	HMI-LAMOL _{TASK} + Real samples	72.2 74.9	72.6 73.9	72.3 75.6	71.8 75.5	72.2 75.0	0.3 0.7
-	Multitask	72.7					

Table 2: Experimental results on the five text classification tasks. (i)–(iv) denote four random task orders. Avg. and Std. respectively represent average and standard deviation for the four orders.

5.2.2 Five Text Classification Tasks

Settings We applied the single-pass setting, which is considered to make it difficult for HMI-LAMOL to memorize training samples.⁶ We tried the three different γ as in the previous experiments (§ 5.2.1). We used the task-specific tokens for all methods except multitask learning. Following previous studies (de Masson d'Autume et al., 2019; Sun et al., 2020a), we report our results for four random task orders. The four orders are shown in Appendix B.2.

Results As shown in Table 2, HMI-LAMOL improved upon LAMOL also in this scenario. For all γ , the p-value of the paired t-test between the results on the four task orders of LAMOL and HMI-LAMOL was smaller than 1%. In particular, the performance gains were larger for smaller γ . Moreover, HMI-LAMOL had smaller standard deviations for the four task orders, which indicates that it is robust to the task training order of CL.

5.2.3 Sample Selection Strategies

Although HMI can control sample generation at the sample level as well as at the task level, in the previous experiments, we randomly selected memory engrams to use for the sample generation for each replay step. In this section, we compare the following three selection strategies with the random selection:

- **Nearest K-means** Inspired by previous studies on real samples replay (Wang et al., 2020; Huang et al., 2021), we compute k-means centroids of all memory engrams in the hippocampal memory for each task, where k is the number of generated re-

Methods	NLU	U tasks Text classification task				asks
Methous	desc	asc	(i)	(ii)	(iii)	(iv)
Random	72.9	69.7	68.7	67.4	67.4	68.4
Nearest K-means	73.1	69.1	69.0	68.6	67.0	69.0
Loss Difference	71.8	65.5	62.4	66.2	67.4	68.0
Low Perplexity	71.8	59.3	67.8	63.7	64.9	68.6

Table 3: Results of HMI-LAMOL with different selection strategies for generating samples.

play samples for each task, and choose the nearest memory engram for each centroid. This strategy can be used to include more diverse samples in replay than the random selection.

- Loss Difference We select samples with a larger loss difference between before and after training $\mathcal{L}_{before} - \mathcal{L}_{after}$. The samples selected through this process are considered more effective for model training.

- **Low Perplexity** To ensure the quality of generated samples, we use memory engrams for the samples with low perplexity of the model after training.

In order to simplify the comparison of the selection strategies, we do not apply the memory pruning (§ 4.2.3) and select samples from all previously learned samples. We also set the small $\gamma = 0.01$, where the difference in the selection strategy is more likely to affect CL performance. We tried all task orders for both CL scenarios with the task-specific tokens.

Results Table 3 shows the results. We first find that the selection strategies clearly affected the final performance of CL. This indicates that the control of the sample generation with HMI-LAMOL also works at the sample level as well as at the task level. This further indicates that HMI-LAMOL has the potential to improve performance when we can use a better selection strategy.

The results indicate that nearest k-means had stably good performance. However, we did not observe a clear advantage compared with random selection. This might be because even random selection can also include a sufficient number of varied samples. In contrast with these strategies focusing on the diversity of generated samples, the other two strategies, which are based on a single measure, such as loss difference or perplexity, did not perform well and lacked stability. After observing generated samples with these strategies, we discovered a serious bias in the generated samples: the selection based on the loss difference included more samples from a single class in the text classifi-

⁶Results in the multiple-pass setting are shown in Appendix D.



Figure 3: Task balance in replay samples generated with LAMOL and HMI-LAMOL in the experiments of the NLU tasks. Each graph shows the results of generated replay samples when training for each new task denoted with [].

cation task; generated samples selected on the basis of the perplexity tended to be of high quality but short in length and easy to predict. In conclusion, our experiments demonstrated that the diversity of the generated samples contributes to CL performance, which is consistent with the recent findings on real samples replay Wang et al. (2020); Huang et al. (2021).

6 Analysis

6.1 Balance of Replay Samples

To validate the controllability in sample generation in HMI-LAMOL, we investigated the balance of each task in the replay samples generated in the experiments on the five NLU tasks, described in § 5.2.1. To classify the replay samples, we used the BERT classifier model trained with the same training data as the experiments of CL.

Figure 3 shows the portion of each task in the generated replay samples for each replay step in the CL experiments. We first find that LAMOL generated many samples from more recently learned tasks rather than from older tasks. It became more evident in the smaller γ , which is consistent with the performance trend in CL. Although task-specific tokens alleviate this problem to some extent, when $\gamma = 0.01$, almost all of the generated samples were from the most recently learned task in all replay steps. These results indicate that the task-specific tokens are insufficient to tie the

	NLU tasks	Text classification tasks
raw text	103,329 KB	340,200 KB
+ gzip	10,879 KB	106,914 KB
no compression	263,553 KB	862,500 KB
+ PQ	3,129 KB	9,368 KB
+ PQ, Pruning	540 KB	540 KB

Table 4: Storage requirements for the hippocampal memory after training the last task in our two CL experiments. *raw text* indicates the size of ASCII text file containing all real samples, and the next line is the size after gzip compression. *no compression* means the case of keeping all samples as real-valued vectors.

generated samples to each task.

However, HMI-LAMOL successfully controls the sample generation for each previous task. In contrast to LAMOL, it enables the generation of samples for older tasks even with the small γ .

6.2 Effect of Memory Compression

HMI uses the feature quantization (§ 4.2.2) and the memory pruning (§ 4.2.3) to reduce the extra storage space required to store memory engrams. Table 4 shows the storage requirements for HMI with and without the two compression methods. It also shows the storage requirements when keeping real samples as raw text file and when applying gzip compression.

When keeping real-valued vectors without the feature quantization, HMI requires even more stor-

age space than when keeping real samples. However, after applying the quantization, each memory engram is compressed to 16 bytes, which is 96x smaller than real-valued vectors and suppressed much less than keeping real samples. Note that the numbers in Table 4 include the PQ codebook, which is 384 KB in size. In addition, the memory pruning reduces it to fixed 540 KB, which the storage requirements will never exceed even if the number of tasks or training samples increase.

7 Conclusion

We proposed hippocampal memory indexing (HMI), inspired by the biological brain, that controls generative replay by conditioning sample generation with compressed representations of previous training samples. Experimental results indicated that HMI successfully controls the sample generation of generative replay and consistently improves the CL performance and robustness. HMI is expected to be further improved by exploring better selection strategies for generating samples.

Limitations

First, in contrast to most existing generative replay approaches, HMI needs extra data space to store features of previous training samples for the sample generation control, while these features are compressed to quantization indices, which require smaller storage space, and their total number is limited to at most M_{max} by the memory pruning.

Second, there is still a performance gap between the replay of generated samples and real samples in HMI. This indicates that it is difficult to completely reconstruct previously learned samples from memory engrams.

Third, although HMI can control the generated samples for replay, there is room for further investigation into the selection strategies better than the random selection. In addition, we tried only random selection and did not further investigate the selection methods for samples in the memory pruning, which can be also explored in future work.

References

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

- Zhiyuan Chen, Nianzu Ma, and Bing Liu. 2015. Lifelong learning for sentiment classification. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 750– 756, Beijing, China. Association for Computational Linguistics.
- Heejeong Choi and Pilsung Kang. 2021. Lifelong language learning with the most forgotten knowledge. *IEEE Access*, 9:57941–57948.
- 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, Online. Association for Computational Linguistics.
- Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. In Advances in Neural Information Processing Systems, volume 32, pages 13122–13131. Curran Associates, Inc.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Le Fang, Tao Zeng, Chaochun Liu, Liefeng Bo, Wen Dong, and Changyou Chen. 2021. Transformerbased conditional variational autoencoder for controllable story generation. *arXiv preprint arXiv:2101.00828*.
- Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. 2020. Wiki-40B: Multilingual language model dataset. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 2440– 2452, Marseille, France. European Language Resources Association.
- Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6440, Online. Association for Computational Linguistics.
- 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, Online. Association for Computational Linguistics.

- Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.
- Kasidis Kanwatchara, Thanapapas Horsuwan, Piyawat Lertvittayakumjorn, Boonserm Kijsirikul, and Peerapon Vateekul. 2021. Rational LAMOL: A rationalebased lifelong learning framework. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 2942–2953, Online. Association for Computational Linguistics.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.
- Sungjin Lee. 2017. Toward continual learning for conversational agents. *CoRR*, abs/1712.09943.
- Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020a. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4678–4699, Online. Association for Computational Linguistics.
- Yuanpeng Li, Liang Zhao, Kenneth Church, and Mohamed Elhoseiny. 2020b. Compositional language continual learning. In *International Conference on Learning Representations*.
- 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, Volume 1 (Long and Short Papers), pages 3274–3279, Minneapolis, Minnesota. Association for Computational Linguistics.
- Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021.
 Continual learning in task-oriented dialogue systems.
 In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7452–7467, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7765–7773.

- Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.
- Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.
- German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. arXiv preprint arXiv:1606.04671.
- Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In Advances in Neural Information Processing Systems, volume 30, pages 2990–2999. Curran Associates, Inc.
- Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020a. Lamol: Language moodeling is for lifelong language learning. In *International Conference on Learning Representations*.
- Jingyuan Sun, Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2020b. Distill and replay for continual language learning. In Proceedings of the 28th International Conference on Computational Linguistics, pages 3569–3579, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Timothy J Teyler and Jerry W Rudy. 2007. The hippocampal indexing theory and episodic memory: updating the index. *Hippocampus*, 17(12):1158–1169.
- Han Wang, Ruiliu Fu, Xuejun Zhang, and Jun Zhou. 2022. Rvae-lamol: Residual variational autoencoder to enhance lifelong language learning. arXiv preprint arXiv:2205.10857, abs/2205.10857.
- Zirui Wang, Sanket Vaibhav Mehta, Barnabas Poczos, and Jaime Carbonell. 2020. Efficient meta lifelonglearning with limited memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 535–548, Online. Association for Computational Linguistics.
- Rui Xia, Jie Jiang, and Huihui He. 2017. Distantly supervised lifelong learning for large-scale social media sentiment analysis. *IEEE Transactions on Affective Computing*, 8(4):480–491.

- Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Lifelong domain word embedding via meta-learning. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 4510–4516. International Joint Conferences on Artificial Intelligence Organization.
- Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 3987– 3995. PMLR.

Methods	Pre-training		Text (i)		
LAMOL	$ \mathcal{L}_{LM}$		71.0 70.0		
HMI-LAMOL	$\begin{array}{l} \mathcal{L}_{AE} \\ \mathcal{L}_{AE} + \mathcal{L}_{LM} \end{array}$	76.2 76.6	71.8 72.2		

Table 5: Results of CL experiments with $\gamma = 0.2$ and using task-specific tokens for the different pre-training tasks.

A Effect of Pre-training

As described in § 4.4, we applied the LM loss as well as the AE loss to the pre-training in HMI. Table 5 shows the CL performance of LAMOL and HMI-LAMOL for the different pre-training tasks.

The results show that HMI-LAMOL pre-trained with LM, outperformed HMI-LAMOL, pre-trained only with AE, for all task settings, which indicates pre-training LM is effective in HMI-LAMOL. Whereas, it is difficult to say that the pre-training LM is also effective in LAMOL, and the performance of LAMOL, pre-trained with LM, was still lower than HMI-LAMOL. Therefore, the gain of HMI is not only due to the pre-training.

B Tasks and Orderings

B.1 Details of Datasets

Following previous studies, we used five NLU tasks and five text classification tasks for our experiments. Table 6 contains a summary of the datasets, their size, metrics, and examples of the SQuAD-like QA scheme in decaNLP (McCann et al., 2018).

B.2 Ordering for Text Classification Tasks

For text classification tasks, we consider four random permutation orders, which mirror those in a previous study (de Masson d'Autume et al., 2019): (i) Yelp \rightarrow AGNews \rightarrow DBPedia \rightarrow Amazon \rightarrow Yahoo, (ii) DBPedia \rightarrow Yahoo \rightarrow AGNews \rightarrow Amazon \rightarrow Yelp, (iii) Yelp \rightarrow Yahoo \rightarrow Amazon \rightarrow DBpedia \rightarrow AGNews, (iv) AGNews \rightarrow Yelp \rightarrow Amazon \rightarrow Yahoo \rightarrow DBpedia.

C Details of Experiment Settings

We used a single Tesla V100 for all experiments. We implemented all methods with half-precision number (16-bit float). All experiments were averaged over two runs with seed = 42, 43. A summary of the training hyperparameter settings, following the settings in the official implementation of LAMOL,³ are listed in Table 7. More details of the implementation can be found in our released experimental source code.¹

D Five Text Classification Tasks with Multiple-pass Setting

In our experiments with the five text classification tasks (§ 5.2.2), we evaluated our HMI in the singlepass setting, which is considered as an ideal setting for CL and where HMI is more difficult to memorize training samples. In this section, we also present the performance of HMI when training for each task for nine epochs, the same setting as in previous work.

Table 8 shows the results of LAMOL and HMI-LAMOL, which use task-specific tokens and $\gamma = 0.2$, and other current methods. The results indicate that HMI-LAMOL also consistently outperformed LAMOL in this setting and narrows the gap with the replay of real samples. Note that although Meta-MbPA also has a good performance, MbPA++ and Meta-MbPA cannot be directly compared to other methods because of using real samples.

Task	Dataset	Context	Question	Answer	# of Train	# of Test	Metric
Question Answering	SQuAD	Albert Einstein lived in a flat at the Kramgasse 49,	Where is Albert Einstein live?	The Kramgasse 49	87,599	10,570	nF1
Semantic Parsing	WikiSQL	The table has columns club, Which club was founded ?	What is the translation from English to SQL?	SELECT player from table WHERE	56,355	15,878	lfEM
Sentiment Analysis	SST	It's a very valuable film	Is this review negative or positive?	positive	6,920	1,821	EM
Semantic Role Labeling	QA-SRL	The trilogy was released on vinyl by ipecac recordings.	What was released on something?	The trilogy	6,414	2,201	nF1
Goal-oriented Dialogue	WOZ	I am looking for African food	What is the change in state?	food: African;	2,536	1,646	dsEM
	AGNews	Smart phone market growing	Is this sentence World, ?	Sci/Tech			
	Yelp	Nothing special, your typical buffet food	Is this sentence very negative, ?	negative			
Text Classification	Amazon	One of the worst comercials	Is this sentence very negative, ?	very negative	115,000	7,600	EM
	DBPedia	Rubyville Elementary School	Is this sentence Company, ?	EducationalInstitution			
	Yahoo	What should I do, I cant quit smoking?	Is this sentence Society & Culture, ?	Health			

Table 6: Summary of datasets, size, metrics, and example of conversion to dacaNLP format of all tasks.

Hyperparameter	Value
optimizer	Adam
epsilon of Adam	$1.0 imes 10^{-4}$
learning rate	$6.25 imes 10^{-5}$
learning rate schedule	warm-up linear
warm-up ratio	0.005
weight decay	0.01
max gradient norm	1.0

Table 7: Training hyperparameters in our experiments.

Methods	(i)	(ii)	(iii)	(iv) Avg.
MbPA++ (de Masson d'Autume et al., 2019)	70.8	70.9	70.2	70.7 70.6 75.5 75.3 77.6 77.3
MbPA++ (Wang et al., 2020)	75.3	74.6	75.6	
Meta-MbPA (Wang et al., 2020)	77.9	76.7	77.3	
LAMOL (Sun et al., 2020a)	76.7	77.2	76.1	76.1 76.5
DnR (Sun et al., 2020b)	77.4	77.2	77.1	76.9 77.2
LAMOL	76.6	76.8	76.8	76.8 76.8
HMI-LAMOL	77.5	77.5	77.8	77.3 77.5
HMI-LAMOL + Real samples	77.5	77.5	78.0	78.0 77.7

Table 8: Results on five text classification tasks in the multiple-pass setting. LAMOL and HMI-LAMOL were evaluated with our implementation (bottom three rows), and the other scores are obtained from each paper.