Distill and Replay for Continual Language Learning

Jingyuan Sun,[1,2] **Shaonan Wang,**[1,2] **Jiajun Zhang,**[1,2,4] **Chengqing Zong**[1,2,3]
[1]National Laboratory of Pattern Recognition, CASIA, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, China
[4]Beijing Academy of Artificial Intelligence, Beijing, China
{jingyuan.sun, shaonan.wang, jjzhang, cqzong}@nlpr.ia.ac.cn

Abstract

Accumulating knowledge to tackle new tasks without necessarily forgetting the old ones is a hallmark of human-like intelligence. But the current dominant paradigm of machine learning is still to train a model that works well on static datasets. When learning tasks in a stream where data distribution may fluctuate, fitting on new tasks often leads to forgetting on the previous ones. We propose a simple yet effective framework that continually learns natural language understanding tasks with one model. Our framework distills knowledge and replays experience from previous tasks when fitting on a new task, thus named DnR (distill and replay). The framework is based on language models and can be smoothly built with different language model architectures. Experimental results demonstrate that DnR outperfoms previous state-of-the-art models in continually learning tasks of the same type but from different domains, as well as tasks of radically different types. With the distillation method, we further show that it's possible for DnR to incrementally compress the model size while still outperforming most of the baselines. We hope that DnR could promote the empirical application of continual language learning, and contribute to building human-level language intelligence minimally bothered by catastrophic forgetting.

1 Introduction

Humans and many advanced animals can learn new tasks without necessarily forgetting the old ones (Glenberg, 1997; Zenke et al., 2017). This ability to continuously learn, accumulate knowledge and reuse them to tackle new challenges through the lifespan is a critical requirement for human-like intelligence. However, the currently dominant paradigm for machine learning is still to train a model on a static dataset, where state-of-the-art methods deliver impressive performance on that particular task. But when learning tasks in a stream where data distribution may shift, the models generally fail to isolate acquired knowledge and forget previously learned tasks. Such phenomenon is known as catastrophic forgetting.

There are mainly two stretches of methods in overcoming catastrophic forgetting. One is data-based method that achieves continual learning through reproducing the data distribution of old tasks (Kamra et al., 2017; Chaudhry et al., 2018). Either storing real or generating pseudo examples, the model replays the training data of experienced tasks when learning a new one. The other stretch is model-based that enables continual learning with modification to the model architecture (Schwarz et al., 2018; Masse et al., 2018). For example, some methods (Lee et al., 2017; Aljundi et al., 2018) regularize the loss function to constrain the updates of weights, especially those important for solving tasks. In the continual learning settings of the above methods, the tasks in a stream are mostly in essence of the same type, just different in domains. And most of the methods have only been applied to solve computer vision tasks.

The emergence and rapid development of large-scale pre-trained language models make it possible to solve essentially different types of natural language understanding (NLU) tasks with one base architecture. But these over-parameterized models are also in risk of over-fitting, more prone to forgetting when sequentially learning diverse data distributions (Parisi et al., 2019). To this end, we propose a framework

Proceedings of the 28th International Conference on Computational Linguistics, pages 3569–3579
Barcelona, Spain (Online), December 8-13, 2020

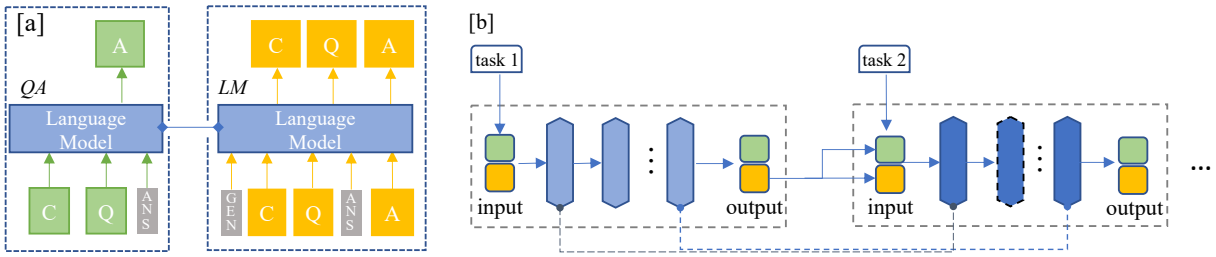

Figure 1: [a] The overview of training DnR on a single task that simultaneously learns to optimize question answering and language modeling. [b] When continually learning along a task sequence, the knowledge is explicitly transferred through distillation and generative replay. The distillation method can also be used for model compression if needed.

that performs generative experience replay and knowledge distillation to allow language models to continually learn and solve different NLU tasks. It is a data-based method. In experience replay, the model generates examples that imitate the data distribution of experienced tasks to retrain on. Inspired by Sun et al. (2020), language models are natural text generators, so no additional generation or memory module are required. In knowledge distillation, the model aligns the semantic space learned on a new task with the old ones and the alignment is simultaneous with the model optimization. The generated examples are discrete while the semantic spaces are continuous. Transferring both the discrete and continuous representations could lead to better isolation and re-usage of learned knowledge for continual learning. With the distillation methods, it is also possible for DnR to compress the base language model size, tackling catastrophic forgetting with an incrementally lighter model.

We compare DnR against strong baselines in learning a stream of tasks, considering two application scenarios of continual learning. In the first setting, the tasks in the sequence are of the same type but from different domains, where we select five text classification datasets. In the second setting, the tasks in the sequence are of different types, where we adopt five tasks including semantic labeling and task-oriented dialogue. In both cases, DnR outperforms previous state-of-art methods. DnR is easy to implement and effective in results, promoting the empirical application of continual language learning. We also hope that DnR could contribute to building human-level language intelligence that is no longer bothered by catastrophic forgetting.

2 Related Work

Tackling new tasks without necessarily losing the knowledge learned in previous tasks is one fundamental requirement for a human-like intelligent system (Parisi et al., 2019). However, the currently dominant paradigm of machine language learning is still training a model on a static dataset to achieve satisfactory performance on that particular task (Wang et al., 2018; Ostapenko et al., 2019). Most of these methods, especially the deep neural network-based ones, do not fare well in the continual learning scenarios. In continual learning, a model is required to fit on a stream of tasks where data distribution may not be uniform. For example, a network fitted on a first task tends to forget how to perform on it after trained on a sequential new task (Sun et al., 2018; Shin et al., 2017). This problem, namely catastrophic forgetting, poses a severe challenge in building a general language intelligent system with lifelong learning capacity.

Efforts have been made in recent years to overcome the catastrophic forgetting of deep neural networks. There are mainly two stretches of methods, differentiated by the way of isolating and reusing the accumulated knowledge. One stretch of method works by reproducing the data distribution of witnessed tasks. Some members of this family select and store informative samples in an explicit memory module. The memorized samples will be re-trained when learning future tasks. For example, MBPA from de Masson d'Autume et al. (2019) uses such an episodic memory module for sparse experience replay and local adaptation to allow continual learning. Due to the efficiency and memory limit of storing raw examples, some methods supplant the explicit memory with a generator. LAMOL (Sun et al., 2020), for example, is trained to simultaneously solve a task and imitate training examples, thus saving

the distribution of previous tasks. The other stretch of method regularizes model structures to remember old tasks while fitting on the new ones. Instead of preserving informative samples, these methods seek to protect important weights of the network for performing a particular task. For example, Elastic Weight Consolidation (EWC) by Kirkpatrick et al. (2016) constrains change of informative weights for previous tasks identified by Fisher's information. Instead of relying on such parametric regularization, Serrà et al. (2018) proposed to split a base network into dedicated parameter subspaces for different tasks to allow continual learning. Worth noting that there are also hybrid approaches that integrate the above two stretches of methods to overcome catastrophic forgetting. Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017) and its imrproved version AGEM (Chaudhry et al., 2018) sample and store a subset of real samples from previous tasks and use them to constrain parameter gradients when optimizing on the following new tasks.

3 Methods

3.1 Data Formatting

To build a single model that continually generalizes across tasks (especially tasks of different types), we need to unify the formation of their datasets. Inspired by DecaNLP (McCann et al., 2018), multiple downstream natural language understanding tasks can be formulated as question answering. For example, sentiment classification on a sentence can be seen as generating an answer (A) to the question (Q) *"Is this sentence sentimentally positive or negative?"* given the sentence as context (C). In this paper, we will test the continual learning ability on five different tasks in the scheme of DecaNLP, following the setting of (Sun et al., 2020). Note that decaNLP supports ten different types of NLU tasks so our model is not constricted to the five selected tasks. As shown in Table 1, the samples of these tasks are framed into the scheme of SQuAD, a classical QA dataset. Then the C, Q, A of each training example are concatenated as a sequence, annotated by some tokens. An [ANS] token will be inserted between Q and A, indicating the starter point of decoding in question answering. What's more, DnR simultaneously optimizes question answering and language modeling (which will be detailed in the following sections). A [GEN] token is inserted at the head of the C-Q-A sequence to further make a copy in a language modeling format. Every example of a dataset will be fed to DnR in these two formats as depicted in Fig.1[a].

Task	Context (C)	Question (Q)	Answer (A)	Dataset	Train	Test	Metric
Sentiment Classification	We love the stirring, funny film.	Is this sentence positive or negative?	positive	SST	6920	1821	EM
Semantic Role Labeling	The race is in mixed eights, and usually held in late February.	When is something held ?	in late February	QA-SRL	6414	2201	nF1
Goal Oriented Dialogue	Are there Eritrean restaurants in town?	What is the change in dialogue state?	food: Eritrean	WOZ	2536	1646	dsEM
Question Answering	… and AFC stands for American Football Conference..	What does AFC stand for>	American Football Conference	SQUAD	87599	10570	nF1
Semantic parsing	The table has column names… Tell me what the notes are for South Australia	What is the translation from English to SQL?	SELECT notes from table WHERE..	WIkiSQL	56355	15878	lfEM

Table 1: Example of the NLU tasks we test that are framed in the context-question-answer format of SQuAD in decaNLP. Here we also describe the train-test split and evaluation metrics of the five tasks.

3.2 Training on a Single Task

DnR is continually trained on a sequence of tasks, $T_1, T_2, ..., T_n$. Beginning from the first task T_1 in the sequence, the model learns to solve the task and generate the training examples of the task. As shown in Fig.1[a] the examples are framed into the SQuAD-like context-question-answer format. Assume that the *n* training examples of task T_1 are $\{S_1^{T_1}, S_2^{T_1}, ...S_i^{T_1}..., S_n^{T_1}\}$, where $S_i^{T_1} = \{C_i^{T_1}, Q_i^{T_1}, A_i^{T_1}\}$. On one hand, the model is optimized as in question answering, predicting answers after reading the context and

question of every training example:

$$p^{QA}(x) = p\left(A_i^{T_1} | C_i^{T_1}, Q_i^{T_1}\right) \tag{1}$$

On the other hand, the model is also optimized as a language model, estimating the distribution of the whole training example as a sequence. Assume that $\{C_i^{T_1}, Q_i^{T_1}, A_i^{T_1}\}$ is concatenated as a sequence of length n_i and s_k denotes one item of this sequence. The model is optimized to maximize

$$p^{LM}(x) = \prod_{i=1}^{n_i} p\left(s_k | s_1, \ldots, s_{k-1}\right). \tag{2}$$

for every training example. We use cross entropy as training objective for both question answering and language modeling, QA for example:

$$\mathbb{L}^{QA} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i^{QA}\log\hat{y}_i^{QA} + \left(1 - y_i^{QA}\right)\log\left(1 - \hat{y}_i^{QA}\right)\right] \tag{3}$$

where $\hat{y}_i^{QA}$ denotes the decoded answer on sample *i* for question answering, while $\hat{y}_i^{LM}$ denotes the predicted token. y_i^{QA} and y_i^{LM} denote the targets. $\mathbb{L}^{QA}$ and $\mathbb{L}^{LM}$ are optimized together as $\mathbb{L}^{Task} = \mathbb{L}^{QA} + \lambda^{LM}\mathbb{L}^{LM}$, where $\mathbb{L}^{LM}$ is a hyper-parameter.

3.3 Distil and Replay

During the process of continual learning, before fitting on the next task, the model first generates pseudo samples by top-k sampling that represent the data distribution of previous tasks. The number of generated examples is $\gamma|T_i|$, where γ is the hyper-parameter of sampling ratio and $|T_i|$ is the size of task T_i's training data. The generated data are replayed into the current dataset and fed to the model, as shown in Fig.1[b]. Then the losses are calculated as in the previous subsection. The generated examples can be seen as a discrete representation of the data distribution of the previous task.

Besides augmenting the current dataset with the generated examples, DnR also performs knowledge distillation with the generated examples. Treating the model trained on a previous task as the *teacher* and model learning the current task as the *student*, we align the hidden states as well as attention matrices by optimizing

$$\mathbb{L}^{dil} = \sum_{l}^{n_l}[(1 - \frac{h_{T_{i-1}}^l \cdot h_{T_i}^l}{||h_{T_{i-1}}^l|| \times ||h_{T_i}^l||}) + a_{T_{i-1}}^l \log\frac{a_{T_{i-1}}^l}{a_{T_i}^l}], \tag{4}$$

where $h_{T_i}^l$ denotes layer l's hidden states of the model trained on task T_i, while $a_{T_i}^l$ denotes the attention matrices. n_l means the number of layers to be distilled. Therefore, optimizing the loss $\mathbb{L}^{dil}$ is to minimize the cosine distance between the hidden states and the KL divergence between attention matrices of the distilled layers. The distillation loss is optimized simultaneously with model training to minimize $\mathbb{L} = \mathbb{L}^{Task} + \lambda^{dil}\mathbb{L}^{dil}$.

For distillation, we respectively perform three different methods to match the internal representations of teacher and student as follows. Methods that the target layers for distillation are fixed and optimized altogether, as well as that the targets are dynamic and optimized progressively, are proposed.

- Naive matching: the selected layers of the student model are simultaneously optimized to match the corresponding layers of the teacher model, as shown in Fig.2[a].
- Incremental matching: rather than distilling the layers simultaneously, in the second method we progressively match the hidden layers from the bottom to the top, inspired by (Aguilar et al., 2020). As shown in Fig.2[b], such distillation is dynamic and the target layer for matching moves from the lower to the higher until the output layer. One layer is optimized at a time and the loss is calculated by the transition ① → ② → ③.

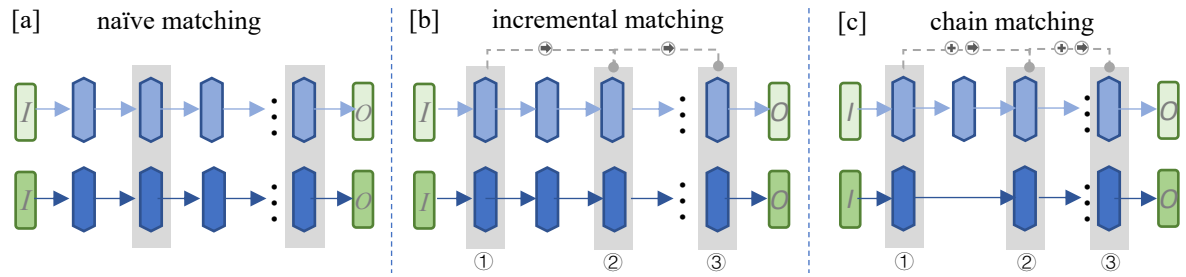

Figure 2: The three distillation methods for matching internal representations. finding dedicated parameter sub-space in the single base network for each subject while allowing them to mutually overlap.

- Chain matching: we also progressively distill the knowledge from lower to higher layers. But instead of moving from one layer to another exclusively, we stack the losses of two successive layers as in the transition ① + ② → ② + ③ → ... . This method is illustrated in Fig.2[c].

3.4 Model Compression

Except for transferring knowledge for continual learning, the distillation techniques can further be used for model compression if needed. We consider two approaches for model compression, according to how much computational efficiency is emphasized in practical application.

- Inner-phase compression: the compression phase is integrated in the training process. We directly drop specific layers of the teacher model to make the student model once learning a new task. Except this, no modification is exerted on the continual learning process of DnR. The distillation is still based on the generated examples of previous tasks as subsection 3.3. So this inner-phase method does not harm the computational efficiency of DnR.

- Outer-phase compression: an independent compression phase is performed after learning every task T_i. In this compression phase, the distillation is not only for transferring knowledge across tasks, but also for producing a compressed copy that mimics the behavior of the initial model. So we use the distillation methods proposed in the previous subsection but based on the real examples of the T_i. The outer-phase method is slower than the inner-phase one, but higher compression quality can be expected.

4 Experimental Setup

4.1 Tasks and Datasets

Following the setting of Sun et al. (2020), we select five different tasks from decaNLP. We use Stanford Question Answering Dataset (SQuAD, Rajpurkar et al. (2016)) for question answering task, Stanford Sentiment Treebank (SST, Radford et al. (2017)) for sentiment analysis, WikiSQL (Zhong et al. (2017)) for Semantic Parsing, English Wizard of Oz (WOZ) for task-oriented dialogue, and QA-SRL (He et al., 2015) for semantic role labeling. As shown in Table 1, the samples of these tasks are framed into the scheme of SQuAD by decaNLP.

The setting of continually learning tasks of different types is challenging. To conduct a fair evaluation, we also compare models on learning tasks of the same type but from different domains in a sequence. We follow de Masson d'Autume et al. (2019)'s setting to use Zhang et al. (2015)'s collection of five text classification tasks, as briefed in Table 2.

4.2 Baselines

We include the following baselines in the evaluation. All the baselines are based on GPT2 with 12 hidden layers (pre-trained GPT-2 from the huggingface transformers is used[1]).

[1]https://huggingface.co/gpt2

Task	Metric	Total Train	Total Test	Dataset	Classes	Dataset	Classes	Train	Test
Text Classfication	EM	575000	38000	Yelp	5	Yahoo	10	115000	7600
				Amazon	5	AGNews	4		
				DBPedia	14				

Table 2: The five text classification datasets. We use the balanced version of all datasets created by Zhang et al. (2015). They randomly sampled 115,000 training examples and 7,600 test examples from each dataset, according to the size of the smallest training and test splits of the five tasks.

- Data-based methods: LAMOL[2], the previous state-of-the-art continual language learning method, is adopted. It generates pseudo examples of previous tasks before learning a new task. LAMOL with and without task-specific tokens (LAMOL-t and LAMOL-g) are both considered.
- Model-based methods: EWC[3] and MAS are adopted. They both estimate the importance of weights for solving a task and condition the updating of important weights.
- Hybrid methods: MBPA and GEM[4] (and its improved version AGEM) are hybrids of data-based and model-based methods. They store training examples for experience replay and parameter adaption. For MBPA, we take its improved version MBPA++ in the evaluation.

Except for the above methods, we also include fine-tuning as a baseline, where the GPT2 model is fine-tuned on the sequence of tasks one after another.

4.3 Implementations

We will first test DnR without model size compression. We respectively implement DnR with the three distillation methods proposed in Section 3. For continual learning, a model is required not only to remember old tasks but also to solve new tasks. The distillation should not force the student to fully replicate the behaviors of the teacher. So in the implementations, we do not match the internal representations between all layers of the teacher and student model. For the naive matching (DnR$_{nm}$), we respectively implement distillation with one, two or three layers, selecting from the last three layers. For both the incremental matching (DnR$_{im}$ and DnR$_{cm}$), we respectively implement distillation with the odd or even number layers.

We will then test DnR in continual learning with model compression. DnR compressed by the inner-phase and outer-phase approach are included in the testing, each with the proposed matching policies for distillation. For outer-phase compression, the epoch running is set to be half of DnR's training epochs.

The compared baselines and our model DnR are trained for 9 epochs and optimized by Adam with 0.01 weight decay. The initial learning rate is 6.25×10^{-5}. For GEM, the sampling rate is 5% of the train data size. For DnR, the $\lambda^{LM} = 0.25$. All the models are implemented with PyTorch and trained on NVIDIA GeForce 2080Ti and Tesla V100. DnR and LAMOL are specifically implemented with half-precision floating number.

5 Results

In this section, we will first give an overview of different models' continual learning ability and evaluate if they are robust to the variation of the task order in a sequence. We then compare specifically how well different models overcome the catastrophic forgetting while examining two hyper-parameters that may largely influence DnR's performance in continual learning. In these two subsections, we observe DnR's advantage over the baselines. In the last subsection, we discuss if the distillation approach of DnR can further compress the model size while still outperform other tested models.

[2]https://github.com/jojotenya/LAMOL
[3]https://github.com/stokesj/EWC
[4]https://github.com/facebookresearch/GradientEpisodicMemory

[a]

	SST SRL WOZ	SST WOZ SRL	SRL SST WOZ	SRL WOZ SST	WOZ SST SRL	WOZ SRL SST	AVG	STD
fine-tuning	50.2	24.7	62.9	31.3	32.8	33.9	39.3	12
EWC	50.6	48.4	64.7	35.5	43.9	39	47	8.7
MAS	36.5	45.3	56.6	31	49.7	30.8	41.6	8.9
GEM	50.4	29.8	63.3	32.6	44.1	36.3	42.8	11
LAMOL_{T}	80	80.7	79.6	78.7	78.4	80.5	79.7	0.8
LAMOL_{G}	79.4	79.9	80.1	78.7	79.8	79	79.5	0.5
$\text{DnR}_{\text{NM}}^{-2}$	**81.4**	**82.1**	**81.2**	**81.1**	80.9	**81.6**	**81.4**	**0.4**
$\text{DnR}_{\text{NM}}^{-1,-3}$	81.3	81.1	80.4	79.8	**81.1**	80.9	80.8	0.6
$\text{DnR}_{\text{IM}}^{\text{even}}$	79.6	80.8	79.4	79.2	79.8	80	79.8	0.6
$\text{DnR}_{\text{CM}}^{\text{even}}$	79.8	80.7	79.7	79	80.3	80	79.9	0.6

[b]

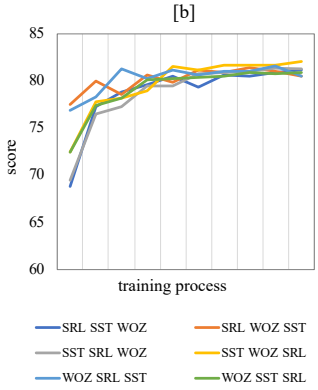

Figure 3: [a] Average testing performance on all the three tasks after learning the last task in the sequence. NM, IM or CM in DnR's subscript denote the distillation approach while the number in the superscript denotes the target layer for matching. [b] Average testing performance of DnR on all the three tasks when learning the last task in the sequence.

order / model	I	II	III	IV	STD	AVG	order / model	I	II	III	IV	STD	AVG
AGEM	70.6	65.9	67.5	63.6	2.94	66.9	LAMOL_g	76.7	**77.2**	76.1	76.1	0.53	76.5
MBPA	74.1	74.9	73.1	74.9	0.85	74.3	DnR	**77.4**	**77.2**	**77.1**	**76.9**	**0.21**	**77.2**

Table 3: Average performance on the five text classification tasks after fitting on the last task in the sequence. I, II, III and IV denote four permutations of the task order.

5.1 Continual Learning

Under the setting of continual learning, one model is required to fit on a sequence of tasks. A robust continual learning model should not deliver starkly different performances when the sequence order changes. To gain a full understanding of the model performance and whether they are influenced by the task order, we first pick three from the five decaNLP tasks to depict, following LAMOL's (Sun et al., 2020) setting. We train every model respectively on all the six permutations of the three tasks. After the model fits on the last task in the sequence, we test it on all the three tasks with the metric demonstrated in Table 1. The three scores are then averaged as a model's continual learning performance. The results are reported in Fig.3[a]. (Note model compression is not implemented unless stated in the experimental results).

We find that fine-tuning, EWC, MAS and GEM are not robust to the variation of task order when learning different types of tasks in a stream. Among the four methods, EWC's performance shows the lowest fluctuation upon the change of task order. But the standard deviation (S.D.) of its performances is still as high as 8.7. Simple fine-tuning turns out to be most largely influenced by the task order. With S.D. 12, its average performance on the sequence "SST-WOZ-SRL" is 63.9% lower than on "SRL-SST-WOZ" where fine-tuning delivers its highest average score. The best performing variation of our model DnR_{nm}^{-2}, exceeds other methods with average score higher than 80 on all the six permutations. Its performance S.D is 0.4, also lower than other baselines, including the previous state-of-art LAMOL. We need to confirm DnR's robustness to the task order. So for each epoch of training the last task, we detail DnR's average performance on all three tasks in Fig.3[b]. We find that at the beginning epochs, DnR in different task orders diverge in performance. But with the training preceding, specifically after 5 epochs, DnR for different task orders gradually converge to a close level around 81.0. This indicates that the task order only exerts minor effects on DnR's continual learning performance.

Continually learning tasks of different types is challenging. For a fair comparison we also test the models on five text classification tasks in a sequence. We follow de Masson d'Autume et al. (2019)'s setting to evaluate on four permutations of the tasks and report the results in Table 3. GEM performs

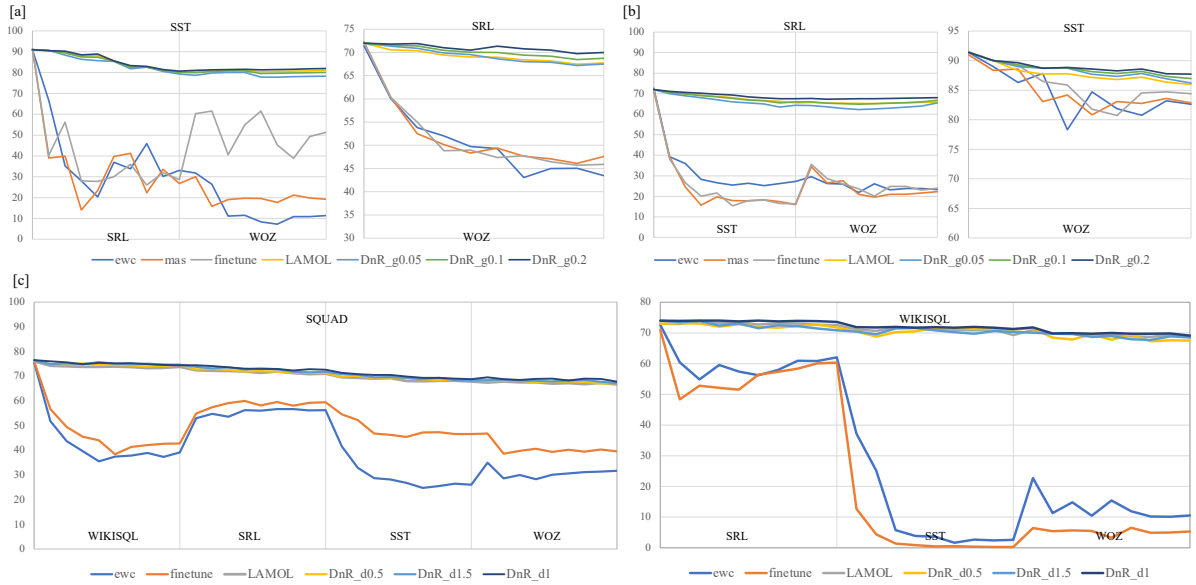

Figure 4: [a] The models' performances on SST and SRL when learning the latter tasks in sequence SST-SRL-WOZ. [b] The models' performances on SRL and SST when learning the latter task in sequence SRL-SST-WOZ. [c] The models' performances on SQuAd and WikiSQl when learning the latter tasks in the sequence SQuAD-WikiSQL-SRL-SST-WOZ. In [a] to [c], horizontal axis represents the training process counted by epoch.

better in this relatively simple setting. DnR still exceeds the baselines in most cases.

5.2 Overcoming Catastrophic Forgetting

In this subsection, we specifically evaluate the models in overcoming catastrophic forgetting of previous tasks when learning a new task. After each epoch of learning the task $T_i, (i > 1)$, we test the model on all the previous tasks $T_1, ..., T_{i-1}$. Specifically, there are two hyper-parameters that may highly influence DnR's performance in overcoming catastrophic forgetting. γ conditions the number of generated pseudo examples and λ^{dil} weights the distillation loss. So we also test DnR^{-2}_{NM} (the best performing variation of DnR as revealed in the previous subsection) with different γ and λ^{dil}, with the best performing variation DnR^{-2}_{NM} as evaluated in the previous section.

We first evaluate DnR with different γ against the baselines. The tested γ values include 0.05, 0.1 and 0.2. In case that the task order may influence some baselines' performance, we follow subsection 5.1 to pick three tasks SST, SRL, WOZ, and respectively test in the order of SST-SRL-WOZ and SRL-SST-WOZ. The results are reported in Fig.4[a] and Fig.4[b]. In Fig.4[a], the left half denotes performance of testing the models on SST when learning SRL and WOZ. The right half denotes testing the models on SRL when learning WOZ. We find that, though starting from a uniform level, the tested models quickly diverge in performances as the continual learning precedes. In the SST-SRL-WOZ sequence, after fitting on SRL, three baselines performance on SST drop more than 50%. After fitting on WOZ, four baselines drop more than 75%. The EWC method suffers most from catastrophic forgetting. Its performance on SST decreases sharply from 90 to 25.4 only after fitting on one new task. DnR outperforms other baselines with all the three tested γ settings. With $\gamma = 0.2$, we observe the best results. Setting of γ influences DnR's performance in overcoming forgetting but not that larger γ yields better results. Actually we do not observe $\gamma = 0.1$ and $\gamma = 0.05$ lead to consistently different performances. Similar patterns are observed for DnR in Fig.4[b]. Changing the task order does not influence DnR's performance as highly as other methods such as EWC.

We then evaluate DnR with different λ^{dil} against the baselines. Here we use all the five tasks in a sequence, namely SQuAD-WikiSQL-SST-SRL-WOZ. We test the model performance on SQuAD and WikiSQL after learning the other tasks in the sequence and show the results in Fig.4[c]. Comparing with

the three-task sequence, the baselines (EWC, MAS and fine-tuning) show visibly larger fluctuations in performance under the five-task setting. For these three models, once they start to learn WikiSQL, we observe that their performances on SQuAD drop sharply. Similar patterns are also observed in the three models' performance on WikiSQL when they learn the WOZ task. Our model, DnR, outperforms these baselines with all three λ^{dil} settings. With $\lambda^{dil} = 1$, DnR achieves its best performance.

5.3 Model Compression

With the distillation method, it's theoretically possible for DnR to incrementally compress the model size during continual learning without largely forgetting the learned knowledge. In this subsection, we will discuss the effects of compressing model size on DnR's continual learning performance. We still pick the three tasks as in the previous sections and demonstrate the findings on the task sequence SST-WOZ-SRL. We report the models' testing performances on previous tasks when learning the last task SRL in Fig.5.

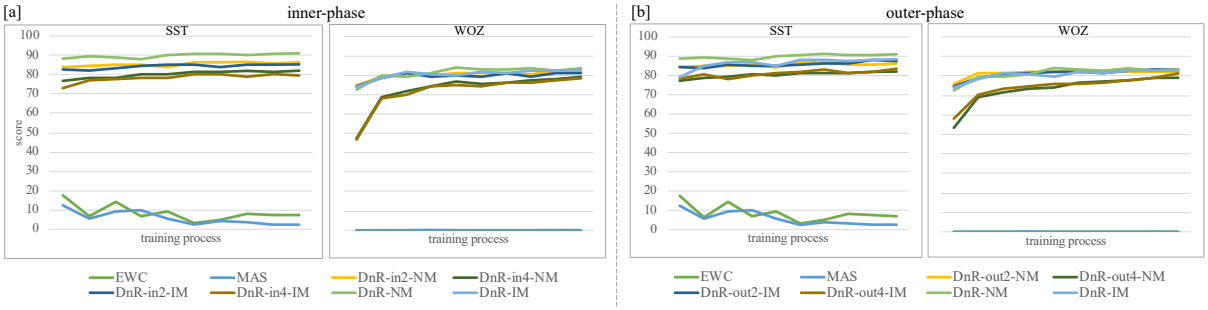

Figure 5: Effect of model compression on DnR's continual learning performance. We compare the models on SST-WOZ-SRL, and depict their performances on the previous tasks when learning the last task SRL. [a] and [b] respectively report DnR with inner-phase and outer-phase compression against the baselines. DnR-in2-NM, for example, denotes dropping two layers with inner-phase compression and naive matching. Horizontal axis represents the training process counted by epoch.

We first find that even with model compressed, DnR still significantly outperforms EWC and MAS in continual learning performance. EWC and MAS suffer from catastrophic forgetting even without any model size compression, especially since learning on WOZ task. We discuss in the previous subsections that WOZ, the task-oriented dialogue task seems to be largely different in data distribution and task requirements with SRL and SST. It might be tough for some baselines to mend the gap. In contrast, dropping two layers of DnR doesn't lead to a stark decrease in its WOZ task performance, especially with the outer-phase compression method. But this doesn't mean that DnR's is not influenced by model compression. With both inner-phase and outer-phase compression, the more layers are dropped, the more that DnR is ablated for continual learning. For example, dropping four layers lesions DnR in remembering SST. Generally, the outer-phase compression method outperforms the inner-phase one. For instance, outer-phase compression yields a lower decrease in DnR's performance on remembering the WOZ task, as revealed by the comparison between Fig.5[a] and Fig.5[b]. It is also interesting to find that incremental-matching is more compatible with outer-phase compression. Incremental matching outperforms normal matching in the outer-phase compression method, but not in the inner-phase one. So if a memory-efficient light-weight model is required in the empirical application of DnR, we tend to recommend the outer-phase compression with incremental matching.

6 Conclusion and Future Work

In this paper, we propose Distill and Replay (DnR), a simple yet effective framework for continual language learning. We perform generative experience replay and knowledge distillation in DnR, transferring both discrete and continuous representations of different task's data distribution to overcome the catastrophic forgetting. DnR is compared against competitive baselines in continually learning tasks of the same type but from different domains, and tasks of radically different types. It outperforms the previous

state-of-the-art method in most cases. We also demonstrate that with the distillation methods it's possible to build incrementally lighter models still capable of continual learning. In future work, we will update the distillation techniques to save the memory cost and optimize the computational efficiency of DnR. We hope DnR could serve as a building block for human-level language intelligence that is minimally bothered by catastrophic forgetting.

Acknowledgements

This work is supported by National Key R&D Program of China 2020AAA0105200 and the Beijing Municipal Science and Technology Project No. Z181100008918017.This work is supported by the Natural Science Foundation of China under Grant 61906189. The research work in this paper is also supported by Beijing Academy of Artificial Intelligence BAAI2019QN0504.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. 2020. Knowledge distillation from internal representations. In *AAAI*, pages 7350–7357.

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.

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

Arthur M Glenberg. 1997. What memory is for. *Behavioral and brain sciences*, 20(1):1–19.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 643–653.

Nitin Kamra, Umang Gupta, and Yan Liu. 2017. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*.

James Kirkpatrick, Razvan Pascanu, Neil C. 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. 2016. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114 13:3521–3526.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. 2017. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, pages 4652–4662.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continuum learning. In *Proceedings of NIPS 2017*.

Nicolas Y Masse, Gregory D Grant, and David J Freedman. 2018. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475.

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*, pages 13122–13131.

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*.

Oleksiy Ostapenko, Mihai Marian Puscas, Tassilo Klein, Patrick Jähnichen, and Moin Nabi. 2019. Learning to remember: A synaptic plasticity driven framework for continual learning. *ArXiv*, abs/1904.03137.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *Proceedings of EMNLP*.

Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*.

Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. *ArXiv*, abs/1801.01423.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. Lamol: Language modeling for lifelong language learning. *Proceedings of the ICLR 2020*.

Jingyuan Sun, Shaonan Wang, and Chengqing Zong. 2018. Memory, show the way: Memory based few shot word representation learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1435–1444.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2018. Associative multichannel autoencoder for multimodal word representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 115–124.

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*, pages 3987–3995. JMLR. org.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.