Meta-Learning with Sparse Experience Replay for Lifelong Language Learning

Anonymous ACL submission

Abstract

Lifelong learning requires models that can continuously learn from sequential streams of data without suffering catastrophic forgetting due to shifts in data distributions. Deep learning models have thrived in the non-sequential learning paradigm; however, when used to learn a sequence of tasks, they fail to retain past knowledge and learn incrementally. We propose a novel approach to lifelong learning of language tasks based on meta-learning with sparse experience replay that directly optimizes to prevent forgetting. We show that under the realistic setting of performing a single pass on a stream of tasks and without any task identifiers, our method obtains state-of-the-art results on lifelong text classification and relation extraction. We analyze the effectiveness of our approach and further demonstrate its low computational and space complexity.

1 Introduction

The ability to learn tasks continuously during a lifetime and with limited supervision is a hallmark of human intelligence. This is enabled by efficient transfer of knowledge from past experience. On the contrary, when current deep learning methods are subjected to learning new tasks in a sequential manner, they suffer from catastrophic forgetting (McCloskey and Cohen, 1989; Ratcliff, 1990; French, 1999), where previous information is lost due to the shift in data distribution. Non-stationarity is inevitable in the real world where data is continuously evolving. Thus, we need to design more robust machine learning mechanisms to deal with catastrophic interference.

Lifelong learning, also known as continual learning (Thrun, 1998), aims at developing models that can continuously learn from a stream of tasks in sequence without forgetting existing knowledge but rather building on the information acquired by previously learned tasks in order to learn new tasks (Chen and Liu, 2018). One conceptualization of this is to accelerate learning by positive transfer between tasks while minimizing interference with respect to network updates (Riemer et al., 2019). Techniques such as regularization (Kirkpatrick et al., 2017) and gradient alignment (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2019) to mitigate catastrophic forgetting have been shown effective in computer vision and reinforcement learning tasks. Meta-learning (Schmidhuber, 1987; Bengio et al., 1991; Thrun and Pratt, 1998) has been applied in continual learning with the objective of learning new tasks continually with a relatively small number of examples per task (Javed and White, 2019; Beaulieu et al., 2020) (in image classification) or in a traditional continual learning setup by interleaving with several past examples from a memory component, i.e. experience replay (Riemer et al., 2019; Obamuyide and Vlachos, 2019a) (in image classification, reinforcement learning and language processing).

In natural language processing (NLP), continual learning still remains relatively unexplored (Li et al., 2020). Despite the success of large pre-trained language models such as BERT (Devlin et al., 2019), they still require considerable amounts of in-domain examples for training on new tasks and are prone to catastrophic forgetting (Yogatama et al., 2019). Existing continual learning approaches to NLP tasks include purely replay-based methods (Wang et al., 2019; Han et al., 2020; d'Autume et al., 2019), a meta-learning based method (Obamuyide and Vlachos, 2019a; Wang et al., 2020) as well as a generative replay-based method (Sun et al., 2020).

Currently, most of the approaches to continual learning employ flawed experimental setups that have blind spots disguising certain weak points. By assuming explicit task identifiers, distinct output heads per task, multiple training passes over the sequence of tasks, and the availability of large

training times as well as computational and memory resources, they are essentially solving an easier problem compared to true continual learning (Farquhar and Gal, 2018). Specifically, while a high rate of experience replay (Lin, 1992) usually mitigates catastrophic forgetting, it comes closer to multi-task learning (Caruana, 1997) than a lifelong learning setup and is computationally expensive when learning on a data stream in real-life applications. Continual learning methods in NLP suffer from these limitations too. An appropriate evaluation of continual learning is thus one where no task identifiers are available, without multiple epochs of training, with a shared output head as well as constraints on time, compute and memory.

In this paper, we propose a novel and efficient approach to lifelong learning on NLP tasks that overcomes the aforementioned shortcomings. We consider the realistic lifelong learning setup where only one pass over the training set is possible with constraints on the rate of experience replay, and no task identifiers are available. Our approach is based on meta-learning and experience replay that is sparse in time and size. We are the first to investigate meta-learning with sparse experience replay in the context of large-scale pre-trained language models, in contrast with previous works that perform replay very often. Additionally, we conduct a systematic study of approaches that rely on pre-trained models and that conform to the same setup.

We extend two algorithms, namely *online meta-learning* (OML) (Javed and White, 2019) and *a neuromodulatory meta-learning algorithm* (ANML) (Beaulieu et al., 2020) to the domain of NLP and augment them with an episodic memory module for experience replay, calling them OML-ER and ANML-ER respectively. While their original objective is to continually learn a new sequence of tasks during testing, we enhance them for the conventional continual learning setup where evaluation is on previously seen tasks. Furthermore, by realizing experience replay as a query set, we directly optimize to prevent forgetting. We show that combining a pre-trained language model such as BERT along with meta-learning and sparse replay produces state-of-the-art performance on lifelong text classification and relation extraction benchmarks when compared against current methods under the same realistic setting. Through further experiments, we demonstrate that BERT combined with OML-ER results in an efficient form of lifelong learning, where most of the weight updates are performed on a single linear layer on top of BERT, while using a limited amount of memory during training and without any network adaptation during test-time. Therefore, our approach is considerably more efficient than previous work in terms of computational complexity as well as memory usage, enabling learning on a task stream without substantial overheads. We hope that our paper informs the NLP community about the right experimental design of continual learning and how meta-learning methods enable efficient lifelong learning with limited replay and memory capacity. To facilitate further research in the field, we make our code publicly available[1].

2 Background and Related Work

Meta-learning In meta-learning, a model is trained on several related tasks such that it can transfer knowledge and adapt to new tasks using only a few examples. The training set is referred to as *meta-training set* and the test set is referred to as *meta-test set*. They consist of *episodes* where each episode corresponds to a task, comprising a few training examples for adaptation called the *support set* and a separate set of examples for evaluation called the *query set*. The goal of meta-learning is to learn to adapt quickly from the support set such that the model can perform well on the query set. Optimization-based meta-learning methods (Finn et al., 2017; Nichol et al., 2018; Triantafillou et al., 2020) have been shown to work well for few-shot learning problems in NLP – specifically machine translation (Gu et al., 2018), relation classification (Obamuyide and Vlachos, 2019b), sentence-level semantic tasks (Dou et al., 2019; Bansal et al., 2019), text classification (Jiang et al., 2018), and word sense disambiguation (Holla et al., 2020).

Continual learning Current approaches to prevent catastrophic forgetting can be grouped into one of several categories: (1) constrained optimization-based approaches with or without regularization (Kirkpatrick et al., 2017; Zenke et al., 2017; Chaudhry et al., 2018; Aljundi et al., 2018; Schwarz et al., 2018) that prevent large updates on weights that are important to previously seen tasks; (2) memory-based approaches (Rebuffi et al., 2017; Sprechmann et al., 2018; Wang et al., 2019;

[1] PlaceholderURL

d'Autume et al., 2019) that replay examples stored in the memory; (3) generative replay-based approaches (Shin et al., 2017; Kemker and Kanan, 2018; Sun et al., 2020) that employ a generative model instead of a memory module; (4) architecture-based approaches (Rusu et al., 2016; Chen et al., 2016; Fernando et al., 2017) that either use different subsets of the network for different tasks or dynamically expand the networks; and (5) hybrid approaches that formulate optimization constraints based on examples in memory (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2019). More recently, Riemer et al. (2019) proposed an approach based on a first-order optimization-based meta-learning algorithm, Reptile (Nichol et al., 2018), augmented with experience replay. However, it involved interleaving every training example with several examples from memory, leading to a high replay rate. Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017) and Averaged-GEM (A-GEM) (Chaudhry et al., 2019) are three popular continual learning methods. EWC introduces a regularization term involving the Fisher information matrix that indicates the importance of each of the parameters to previous tasks. GEM solves a constrained optimization problem as a quadratic program involving gradients from all examples from previous tasks. A-GEM is a more efficient version of GEM since it solves a simpler constrained optimization problem based on gradients from randomly drawn samples from previous tasks in the memory.

Continual learning in NLP Wang et al. (2019) propose an alignment model named EA-EMR that limits the distortion in the embedding space in an LSTM-based (Hochreiter and Schmidhuber, 1997) architecture for lifelong relation extraction. For the same task, Obamuyide and Vlachos (2019a) show that utilizing Reptile (Nichol et al., 2018) with memory can improve performance and call their method MLLRE. Han et al. (2020) further improve relation extraction with their model, EMAR, through episodic memory activation and reconsolidation. d'Autume et al. (2019) propose a model with episodic memory called MbPA++ which incorporates sparse experience replay during training and local adaptation on K-nearest neighbors from the memory for every example during inference. Through their experiments on sequential learning on multiple datasets of text classification and question answering with BERT, they show that their model can effectively reduce catastrophic forgetting. Meta-MbPA (Wang et al., 2020) incorporates local adaptation during meta-training and performs fewer local adaptation steps during testing. Sun et al. (2020) present a model based on GPT-2 (Radford et al., 2019), called LAMOL, that simultaneously learns to solve new tasks and to generate pseudo-samples from previous tasks for replay. They perform sequential learning on five tasks from decaNLP (McCann et al., 2018) as well as multiple datasets for text classification.

All these methods are not yet well-suited for application in real-life scenarios – MbPA++ has slow inference, Meta-MbPA has a higher rate of sampling examples from memory, and other methods require task identifiers and multiple epochs of training. Our approach, on the other hand, alleviates all these problems.

3 Methods

3.1 Task formulation

A typical continual learning setup consists of a stream of K tasks $\mathcal{T}_1$, $\mathcal{T}_2$, ..., $\mathcal{T}_K$. For supervised learning tasks, every task $\mathcal{T}_i$ consists of a set of data points x_j with labels y_j, i.e., $\{(x_j, y_j)\}_{j=1}^{N_i}$ that are locally i.i.d., where N_i is the size of task $\mathcal{T}_i$. We consider the setting where the goal is to learn a function $f_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ by only making one pass over the stream of tasks and with no identifiers of tasks $\mathcal{T}_i$ available. In multi-task learning, on the other hand, it is possible to draw samples i.i.d from all tasks along with training for multiple epochs. Therefore, multi-task learning is an upper bound to continual learning in terms of performance.

We propose an approach to continual learning with meta-learning and experience replay where the updates are similar to first-order MAML. We maintain an episodic memory (or simply called memory) $\mathcal{M}$ which stores previously seen examples. Episodes for meta-training are constructed from the stream of examples as well as randomly sampled examples from $\mathcal{M}$. We perform experience replay sparsely, i.e., a small number of examples are drawn from $\mathcal{M}$ and only after seeing many examples from the stream (i.e. at long intervals), therefore being computationally inexpensive.

3.2 Episode generation and experience replay

We assume that data points arrive in mini-batches of a given size b and every data point has a probabil-

ity p_{write} of being written into an episodic memory module $\mathcal{M}$. We construct episodes on-the-fly from the stream of mini-batches. Given a buffer size m, we construct episode i by taking m mini-batches as the support set $\mathcal{S}_i$ and the next batch as the query set $\mathcal{Q}_i$.

We explicitly define our experience replay mechanism as consisting of two fixed hyperparameters – replay interval R_I, which indicates the number of data points seen between two successive draws from memory, and replay rate $r \in [0, 1]$ which indicates the proportion of examples to draw from memory relative to R_I. Thus, after every R_I examples from the stream, $\lfloor r \cdot R_I \rfloor$ examples are drawn from the memory.

We use these sampled examples from memory as the query set. To perform experience replay in an episodic fashion, we compute the replay frequency R_F as follows (see Appendix A.6):

$$R_F = \left\lceil \frac{R_I/b + 1}{m + 1} \right\rceil \tag{1}$$

Hence, every R_F episodes, we draw a random batch of size $\lfloor r \cdot R_I \rfloor$ from $\mathcal{M}$ as the query set. For other episodes, the query set is obtained from the data stream. The support set for replay episodes is still constructed from the stream. A high r and/or a low R_I ensures that information is not forgotten, but in order to be computationally efficient and adhere to continual learning, it is necessary that r is low and R_I is high, so that the replay is sparsely performed, both in terms of size and time.

During meta-testing, we randomly draw m batches from the memory as the support set and take the entire test set of the respective task as the query set for evaluation. This is done primarily in order to match the testing and training conditions (Vinyals et al., 2016).

3.3 Meta-learning methods

OML-ER The original OML algorithm (Javed and White, 2019) was designed to solve new continual learning problems during meta-testing. Here, we extend it to our setup by augmenting it with an episodic memory module to perform experience replay (ER), and refer to it as OML-ER.

The model $f_{\boldsymbol{\theta}}$ is composed of two functions – a representation learning network (RLN) $h_{\boldsymbol{\phi}}$ with parameters $\boldsymbol{\phi}$ and a prediction learning network (PLN) $g_{\boldsymbol{W}}$ with parameters $\boldsymbol{W}$ such that $\boldsymbol{\theta} = \boldsymbol{\phi} \cup \boldsymbol{W}$ and $f_{\boldsymbol{\theta}}(x) = g_{\boldsymbol{W}}(h_{\boldsymbol{\phi}}(x))$ for an input x. In each episode, the RLN is frozen while the PLN is fine-tuned during the inner-loop optimization. In the outer-loop, both the RLN and the PLN are meta-learned.

During the inner-loop optimization in episode i, the PLN is fine-tuned on the support set mini-batches $\mathcal{S}_i$ with SGD to give:

$$\boldsymbol{W}_i' = \text{SGD}\left(\mathcal{L}_i, \boldsymbol{\phi}, \boldsymbol{W}, \mathcal{S}_i, \alpha\right) \tag{2}$$

where $\mathcal{L}_i$ is the loss function. Using the query set, the objective we optimize for is:

$$J(\boldsymbol{\theta}) = \mathcal{L}_i\left(\boldsymbol{\phi}, \boldsymbol{W}_i', \mathcal{Q}_i\right) \tag{3}$$

During a regular episode, the objective encourages generalization to unseen data whereas during a replay episode, it promotes retention of knowledge from previously seen data.

For the outer-loop optimization, we use the Adam optimizer (Kingma and Ba, 2015) with a learning rate β to update all parameters – both the RLN and PLN:

$$\boldsymbol{\theta} \leftarrow \text{Adam}(J(\boldsymbol{\theta}), \beta) \tag{4}$$

The above optimization would involve second-order gradients. Instead, we use the first-order variant where the gradients are taken with respect to $\boldsymbol{\theta}_i' = \boldsymbol{\phi} \cup \boldsymbol{W}_i'$.

We use $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019) as the RLN (fully fine-tuned; output from the [CLS] token) and a single linear layer mapping to the classes as the PLN.

ANML-ER Beaulieu et al. (2020) proposed ANML that outperformed OML in solving new continual learning problems in image classification. Inspired by neuromodulatory processes in the brain, they design a context-dependent gating mechanism to achieve selective plasticity, i.e., limited and/or selective modification of parameters with new data. We refer to our extension of this method as ANML-ER.

The model $f_{\boldsymbol{\theta}}$ is composed of two networks – a regular prediction network (PN) and a neuromodulatory network (NM) that selectively gates the internal activations of the prediction network via element-wise multiplication. Formally, the NM is a function $h_{\boldsymbol{\phi}}$ with parameters $\boldsymbol{\phi}$, and the PN is a composite function $g_{\boldsymbol{W}_2} \circ e_{\boldsymbol{W}_1}$ with parameters $\boldsymbol{W} = \boldsymbol{W}_1 \cup \boldsymbol{W}_2$. The output is obtained as:

$$f_{\boldsymbol{\theta}}(x) = g_{\boldsymbol{W}_2}\left(e_{\boldsymbol{W}_1}(x) \cdot h_{\boldsymbol{\phi}}(x)\right) \tag{5}$$

In the inner-loop, the NM is fixed while the PN is fine-tuned on the support set. As per our notation, Equation 2 is the form of the inner-loop here too. In the outer-loop, both the NM and the PN are updated as in Equation 4 with first-order gradients.

Our PN is the $\text{BERT}_{\text{BASE}}$ encoder followed by a linear layer mapping to the classes as in OML-ER. For the NM, we use $\text{BERT}_{\text{BASE}}$ followed by two linear layers (768 units) with ReLU non-linearity between them and a final sigmoid non-linearity to limit the gating signal to $[0, 1]$. We keep the NM BERT frozen throughout to reduce the total number of parameters. Our preliminary experiments indicated that fine-tuning the NM BERT in addition produces negligible improvements.

3.4 Baselines

We consider four BERT-based baselines to evaluate the effectiveness of our approach.

SEQ We train our model "traditionally" on all tasks in a sequential manner i.e., one after the other, without replay.

REPLAY It is an extension of SEQ that incorporates sparse experience replay. After seeing R_I examples from the stream, i.e., a replay frequency $R_F = \lceil R_I/b \rceil$, $\lfloor r \cdot R_I \rfloor$ examples are randomly drawn from the memory and one gradient update is performed on them.

A-GEM It requires replay at every training step and task identifiers by default (Chaudhry et al., 2019), but we adapt it to our setting by randomly sampling data points from the memory in sparse intervals.

MTL We train our model in a "traditional" multi-task setup for multiple epochs on mini-batches that are sampled i.i.d from the pool of all tasks. Thus, it serves as an upper bound for the performance of continual learning methods.

4 Experimental setup

4.1 Datasets

Text classification We use the lifelong text classification benchmark introduced by d'Autume et al. (2019) which consists of five datasets[2] from Zhang et al. (2015), trained on sequentially. The datasets are AGNews (news classification; 4 classes), Yelp (sentiment analysis; 5 classes), Amazon (sentiment analysis; 5 classes), DBpedia (Wikipedia article classification; 14 classes) and Yahoo (questions and answers categorization; 10 classes). Following d'Autume et al. (2019), we merge the classes of Yelp and Amazon and have a total of 33 classes, and randomly sample $115,000$ training examples and $7,600$ test examples from each of the datasets since each of them have different sizes. The evaluation metric is the macro average of the accuracies over the five datasets.

Relation extraction We use the lifelong relation extraction benchmark created by Wang et al. (2019) based on the few-shot relation classification dataset FewRel (Han et al., 2018). It consists of $44,800$ training sentences and $11,200$ test sentences, and a total of 80 relations along with their corresponding names available. Each sentence has a ground-truth relation as well as a set of 10 negative candidate relations. The goal is to predict the correct relation among them. To construct tasks for continual learning, they first perform K-means clustering over the average GloVe embeddings (Pennington et al., 2014) of the relation names to obtain 10 disjoint clusters. Each task then comprises of data points having ground-truth relations from the corresponding cluster. In any given task, the candidate relations that were not seen in earlier tasks are removed. But, if all the candidate relations are unseen, the last two candidates are retained. The evaluation metric is the accuracy on a single test set containing relations from all the clusters.

4.2 Implementation

For text classification, we largely maintain the experimental setup of d'Autume et al. (2019). We consider four orders of the datasets (see Appendix A.2) and report the average results obtained from three independent runs. We also set $p_{write} = 1$. While they perform replay by drawing 100 examples from memory for every $10,000$ examples from the stream, we draw 96 examples from memory for every $9,600$ examples which is more convenient with batch size $b = 16$. Thus, we have $r = 0.01$ and $R_I = 9600$. We obtain the best hyperparameters by tuning on the first order of the datasets only. The learning rate for SEQ, A-GEM, REPLAY and MTL is $3\text{e}{-5}$. MTL is trained for 2 epochs. For OML-ER, the inner-loop and outer-loop learning rates are $1\text{e}{-3}$ and $1\text{e}{-5}$ respectively whereas for ANML-ER, they are $3\text{e}{-3}$ and $1\text{e}{-5}$ respectively. The support set buffer size m for both of them is 5. We truncate the input sequence length to 300 for ANML-ER and 448 for the rest. The loss function

[2] https://tinyurl.com/y89zdadp

Method	Accuracy
MbPA++ (d'Autume et al., 2019)	70.6
MbPA++ (Sun et al., 2020)	74.2
LAMOL (Sun et al., 2020)	76.5
Meta-MbPA (Wang et al., 2020)	77.3
SEQ	20.8 ± 0.5
A-GEM	21.9 ± 0.3
REPLAY	67.3 ± 0.7
OML-ER	75.7 ± 0.4
ANML-ER	$\mathbf{75.7 \pm 0.1}$
MTL	79.4 ± 0.2

Table 1: Test set accuracy on text classification.

Method	Accuracy
EA-EMR (Wang et al., 2019)	56.6
MLLRE (Obamuyide and Vlachos, 2019a)	60.2
EMAR (Han et al., 2020)	66.0
SEQ	48.1 ± 3.2
A-GEM	45.5 ± 2.1
REPLAY	65.4 ± 1.2
OML-ER	$\mathbf{69.5 \pm 0.5}$
ANML-ER	68.5 ± 0.7
MTL	85.7 ± 1.1

Table 2: Test set accuracy on relation extraction.

is the cross-entropy loss across the 33 classes. For the evaluation of meta-learning methods, we construct five episodes at meta-test time, one for each of the datasets, where their query sets consist of the test sets of these datasets.

For relation extraction, we consider five orders of the tasks as in Wang et al. (2019). We report the average accuracy on the test set over the five orders, averaged over three independent runs. Sentence-relation pairs are concatenated with a [SEP] token between them to serve as the input. Since this is a smaller dataset, we set $R_I = 1600$ and $r = 0.01$. Additionally, $b = 4$, $m = 5$ and $p_{write} = 1$. Hyperparameter tuning is performed only on the first order. The learning rate is $3\text{e}{-5}$ for SEQ, A-GEM and REPLAY. MTL is trained with a learning rate of $1\text{e}{-5}$ for 3 epochs. The inner-loop and outer-loop learning rates are $1\text{e}{-3}$ and $3\text{e}{-5}$ for OML-ER as well as ANML-ER. All models are trained using the binary cross-entropy loss, treating the true sentence-relation pairs as the positive class and the incorrect pairs as the negative class. The prediction is obtained as an argmax over the logit scores. Meta-learning methods are evaluated using a single meta-test episode with the test set as the query set.

5 Experiments and results

Text classification We present the average accuracy across the baselines and our models with standard deviations across runs in Table 1. We perform significance testing with a two-tailed paired t-test at a significance level of 0.05. Simply training on the datasets sequentially leads to extreme forgetting as reflected in the low accuracy of the SEQ model. With A-GEM, we get only a small, but significant gain ($p = 0.008$) compared to sequential training. By analyzing the frequency of constraint violations in Appendix A.12, we find that A-GEM updates on BERT often behave similar to that in SEQ, which explains its poor performance. REPLAY, on the other hand, drastically improves performance, indicating that BERT benefits substantially even from a sparse experience replay. MbPA++ is the current state-of-the-art on this benchmark under the realistic setup of excluding task identifiers, using sparse replay and a single training epoch. Sun et al. (2020) re-implement MbPA++ and obtain a higher score than the original implementation. We surmise that this is partly attributed to the fact that they perform replay after every 100 steps along with dynamic batching and therefore likely resulting in a higher replay interval. ANML-ER achieves the highest accuracy, demonstrating that our meta-learning setup is more effective at mitigating catastrophic forgetting. OML-ER is almost as effective as ANML-ER, with the differences between the two being statistically insignificant ($p = 0.993$). Although LAMOL has a higher score, it is not directly comparable to our methods since it uses task identifiers and multiple epochs of training, and has a higher generative replay rate of 20%, all of which make the task easier. Meta-MbPA is not directly comparable either since it performs local adaptation on nearest neighbors obtained from the memory during all its inner loop updates, thus having a higher replay rate effectively. Our meta-learning approach further narrows the gap with the MTL upper bound.

Relation extraction We report the average test set accuracy along with the standard deviation across the three runs in Table 2. We see that A-GEM performs similar to SEQ, with the differences being statistically insignificant ($p = 0.218$). Including sparse experience replay (REPLAY) again leads to a substantial increase in performance compared to SEQ. A low A-GEM performance compared to a simple replay method on this benchmark was

also observed in Wang et al. (2019). OML-ER and ANML-ER significantly outperform all the baselines ($p = 0.006$ for OML-ER and $p = 0.026$ for ANML-ER when compared to REPLAY), and the former achieves the highest accuracy overall but, again, the differences between the two are not statistically significant ($p = 0.098$). Although not directly comparable, both of them outperform the previous state-of-the-art LSTM-based method EMAR (Han et al., 2020), despite it using task identities as additional information and training for multiple epochs. There is, however, a wide gap between OML-ER and the MTL upper bound. We return to this in a later analysis.

6 Analysis

Ablation study To investigate the relative strengths of the various components in our approach, we perform an ablation study and report the results in Table 3. Meta-learning without replay leads to a large drop in performance, showing that experience replay, despite being sparse, is crucial. Interestingly however, meta-learning without replay still has considerably higher scores compared to SEQ, demonstrating that it is more resilient to catastrophic forgetting. Retrieving relevant examples from memory and fine-tuning on them during inference is a key aspect in MbPA++ since retrieving random examples instead produces only about 0.4% improvement over REPLAY (d'Autume et al., 2019). Our approach works with random examples and yet achieves substantially higher accuracies. For OML-ER, omitting fine-tuning altogether at the meta-testing stage produces a small, yet significant drop ($p = 0.019$) for text classification, but an insignificant one ($p = 0.602$) for relation extraction. Similarly, for ANML-ER, no meta-test fine-tuning results in a small, significant drop ($p = 0.006$) for text classification and an insignificant change ($p = 0.265$) for relation extraction. Unlike MbPA++, our methods, overall, work well even without additional adaptation steps during inference. Without neuromodulation, ANML-ER is equivalent to standard MAML enhanced with experience replay, which we could call MAML-ER. The performance difference between ANML-ER and MAML-ER is not statistically ($p = 0.120$ for text classification and $p = 0.087$ for relation extraction), which suggests that the neuromodulator in ANML-ER is not useful for our language tasks. Even though OML-ER, ANML-ER and MAML-ER are equally successful in terms of performance, OML-ER is computationally more efficient as only its PLN (a single linear layer) is fine-tuned in the inner-loop.

Method	Accuracy: Text classification	Accuracy: Relation extraction
OML-ER	75.7 ± 0.4	69.5 ± 0.5
– Replay	24.6 ± 0.6	55.9 ± 0.9
– Meta-test fine-tuning	75.6 ± 0.4	69.3 ± 0.7
ANML-ER	75.7 ± 0.1	68.5 ± 0.7
– Replay	51.7 ± 1.8	57.0 ± 0.9
– Meta-test fine-tuning	74.9 ± 0.3	67.7 ± 0.9
– Neuromodulation	75.8 ± 0.2	68.0 ± 0.4

Table 3: Ablation study on model components.

Replay rate	Method	Accuracy: Text classification	Accuracy: Relation extraction
1 %	REPLAY	67.3 ± 0.7	65.4 ± 1.2
	OML-ER	75.7 ± 0.4	69.5 ± 0.5
2 %	REPLAY	67.2 ± 2.0	67.1 ± 0.8
	OML-ER	75.6 ± 0.1	71.6 ± 1.1
4 %	REPLAY	70.3 ± 1.3	69.2 ± 2.2
	OML-ER	76.0 ± 0.6	75.5 ± 0.4
—	MTL	79.4 ± 0.2	85.7 ± 1.1

Table 4: Test metrics on text classification and relation extraction for varying replay rates.

Effect of replay rate We noted previously that there exists a gap in performance between our best model and MTL. To analyze if increasing the replay rate can help narrow the gap, we train both REPLAY and OML-ER with a 2% and 4% replay rate[3], keeping R_I the same as before (Table 4). On text classification, OML-ER has similar performance ($p = 0.936$) with 2% replay rate and a small, significant improvement ($p = 0.001$) with 4% replay rate. The same trend is observed with REPLAY as the replay rate increases ($p = 0.861$ and $p = 0.045$) . In contrast, OML-ER and REPLAY improve by a significantly greater extent on relation extraction ($p = 3\text{e}{-4}$ and $p = 0.048$ respectively). We surmise this is because text classification has equally sized tasks whereas the tasks in relation extraction are imbalanced (see Appendix A.3). Since we employ uniform sampling for memory read/write, this imbalance is reflected in the memory, causing larger tasks to be replayed more

[3]The maximum replay rate for our meta-learning methods is $1/m = 20\%$ i.e., replay every episode with $m = 5$

Memory capacity	Accuracy	
	Text classification	Relation extraction
100 %	75.7 ± 0.4	69.5 ± 0.5
5 %	75.2 ± 0.4	69.2 ± 0.9
1 %	75.6 ± 0.3	66.8 ± 0.3

Table 5: Variation of performance of OML-ER with the size of the memory.

often and underrepresented tasks to be forgotten more quickly. A higher replay rate therefore increases the chances of sufficiently revisiting all previous tasks, leading to better scores. Additionally, on both benchmarks, OML-ER outperforms REPLAY even with higher replay rates. There is still a wide gap between OML-ER with 4% replay and MTL, indicating there is scope for improvement.

Effect of memory size In our experiments so far, we store all the examples in memory; however, this does not scale well when the number of tasks is very large. In order to investigate the effect of memory size on performance, we present the accuracy of OML-ER with 5% and 1% memory capacity in Table 5. We achieve this by setting p_{write} to 0.05 and 0.01 respectively. There are insignificant changes ($p = 0.074$ and $p = 0.952$ respectively) in average accuracy even with reduced memory for text classification. MbPA++, on the other hand, was shown to have a drop of 3% accuracy with 10% memory capacity (d'Autume et al., 2019), which demonstrates that our method is more memory-efficient. Performance on relation extraction suffers a small but significant drop ($p = 0.040$) with 1% memory. The difference is insignificant ($p = 0.741$) with 5% memory and, overall, can still be considered memory-efficient.

Episodic updates In addition to the automatic gradient alignment that comes with meta-learning, we believe that its episodic nature is another reason for its strength in lifelong learning. In text classification for example, SEQ has a replay every 600 optimizer steps whereas meta-learning, by way of its formulation, has a replay every 101 meta-optimizer steps (using Equation 1 with our hyperparameters)[4]. Fewer updates between replays likely aids in knowledge retention. To probe deeper, we trained our REPLAY model such that replay occurs every 100 optimizer steps by setting $R_I = 1600$, with everything else being the same. This achieves an accuracy of 74.4 ± 0.2. Although this is now closer to our meta-learning methods, it is still significantly lower ($p = 0.001$ for OML-ER and $p = 8\text{e}{-5}$ for ANML-ER). Therefore, episodic updates in meta-learning are an important part of the model, contributing positively to performance. For "regular" training to match the same level of performance, experience replay would need to be performed more often.

[4]However, we note that optimizer steps and meta-optimizer steps are not the same nor directly comparable as such.

7 Discussion and conclusion

Continual learning methods so far have relied on manual heuristics and/or have computational bottlenecks. MbPA++ is inexpensive during training due to sparse replay, but its inference is expensive since it requires retrieval of K nearest neighbors for every test example and multiple gradient steps on them. A-GEM, on the other hand, is slower to train due to its projection steps. OML-ER achieves the best of both worlds – its training is fast because its inner-loop, which makes up a large portion of the training, involves only updating the small PLN, and its inference is fast since it relies only on a small number of updates on randomly drawn examples from memory. Furthermore, it also retains its performance when the memory capacity is lowered.

Our method uses a simple, random write mechanism. Other strategies such as those based on surprise (Ramalho and Garnelo, 2019) and forgetting (Toneva et al., 2019) could further refine performance. Furthermore, the problem of task size imbalance could be mitigated with class-balancing reservoir sampling (Chrysakis and Moens, 2020).

In our experiments on text classification, we assume that all the classes are known beforehand. Lifelong learning when the classes are unknown *a priori* and available only during each of the individual tasks is more challenging and would be an interesting extension.

In conclusion, we showed that pre-trained transformer-based language models, meta-learning and sparse experience replay produce a synergy that improves lifelong learning on language tasks in a realistic setup. This is an important step in moving away from manually-designed solutions into simpler, more generalizable methods to ultimately achieve human-like learning. Meta-learning could further be exploited for the combined setting of few-shot and lifelong learning. It might also be promising in learning distinct NLP tasks in a curriculum learning fashion.

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.

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2019. Learning to few-shot learn across diverse natural language classification tasks. *arXiv preprint arXiv:1911.03863*.

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. 2020. Learning to continually learn. *arXiv preprint arXiv:2002.09571*.

Y. Bengio, S. Bengio, and J. Cloutier. 1991. Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 969 vol.2–.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *The European Conference on Computer Vision (ECCV)*.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*.

Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. 2016. Net2net: Accelerating learning via knowledge transfer. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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

Aristotelis Chrysakis and Marie-Francine Moens. 2020. Online continual learning from imbalanced data. In *Proceedings of Machine Learning and Systems 2020*, pages 8303–8312.

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 32*, pages 13143–13152.

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.

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1192–1197, Hong Kong, China. Association for Computational Linguistics.

Sebastian Farquhar and Yarin Gal. 2018. Towards Robust Evaluations of Continual Learning. *Lifelong Learning: A Reinforcement Learning Approach Workshop at ICML*.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

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.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nithin Holla, Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2020. Learning to learn to disambiguate: Meta-learning for few-shot word sense disambiguation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4517–4533, Online. Association for Computational Linguistics.

Khurram Javed and Martha White. 2019. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems 32*, pages 1820–1830.

Xiang Jiang, Mohammad Havaei, Gabriel Chartrand, Hassan Chouaib, Thomas Vincent, Andrew Jesson, Nicolas Chapados, and Stan Matwin. 2018. Attentive task-agnostic meta-learning for few-shot text classification. In *The Second Workshop on MetaLearning at NeurIPS*.

Ronald Kemker and Christopher Kanan. 2018. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Yuanpeng Li, Liang Zhao, Kenneth Church, and Mohamed Elhoseiny. 2020. Compositional language continual learning. In *International Conference on Learning Representations*.

Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. In *Machine Learning*, pages 293–321.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30*, pages 6467–6476.

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 - Advances in Research and Theory*, 24(C):109–165.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? *arXiv preprint arXiv:2004.14448*.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Abiola Obamuyide and Andreas Vlachos. 2019a. Meta-learning improves lifelong relation extraction. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 224–229, Florence, Italy. Association for Computational Linguistics.

Abiola Obamuyide and Andreas Vlachos. 2019b. Model-agnostic meta-learning for relation classification with limited supervision. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5873–5879, Florence, Italy. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Tiago Ramalho and Marta Garnelo. 2019. Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *International Conference on Learning Representations*.

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

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

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*.

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*.

Jurgen Schmidhuber. 1987. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. In *ICML*, pages 4535–4544.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems 30*, pages 2990–2999.

P. Sprechmann, S. Jayakumar, J. Rae, A. Pritzel, A. Puigdomènech, B. Uria, O. Vinyals, D. Hassabis, R. Pascanu, and C. Blundell. 2018. Memory-based Parameter Adaptation. In *ICLR 2018*.

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

Sebastian Thrun. 1998. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer.

Sebastian Thrun and Lorien Pratt. 1998. *Learning to Learn*. Kluwer Academic Publishers, USA.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 2019. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. 2020. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019. Sentence embedding alignment for lifelong relation extraction. 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 796–806, Minneapolis, Minnesota. Association for Computational Linguistics.

Zirui Wang, Sanket Vaibhav Mehta, Barnabas Poczos, and Jaime Carbonell. 2020. Efficient meta lifelong-learning 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.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373.

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, International Convention Centre, Sydney, Australia. PMLR.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657.

A Appendix

A.1 Meta-learning

Optimization-based methods for meta-learning explicitly include generalizability in their objective function and optimize for the same. Model-agnostic meta-learning (MAML) algorithm (Finn et al., 2017) is an optimization-based method that seeks to train a model's initial parameters such that it can perform well on a new task after only a few gradient steps. During meta-training, it involves a two-level optimization process where task adaptation is performed using the support set in an *inner-loop* and meta-updates are performed using the query set in an *outer-loop*. Specifically, parameters $\boldsymbol{\theta}$ of the model $f_{\boldsymbol{\theta}}$ are updated to $\boldsymbol{\theta}'_i$ for task $\mathcal{T}_i$ in the inner-loop by m steps of gradient-based update U on the support set as:

$$\boldsymbol{\theta}'_i = U(\mathcal{L}^s_{\mathcal{T}_i}, \boldsymbol{\theta}, \alpha, m) \tag{6}$$

where $\mathcal{L}^s_{\mathcal{T}_i}$ is the loss on the support set and α is the inner-loop learning rate. The outer-loop objective is to have $f_{\boldsymbol{\theta}'_i}$ generalize well across tasks from a distribution $p(\mathcal{T})$:

$$J(\boldsymbol{\theta}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^q_{\mathcal{T}_i}(f_{U(\mathcal{L}^s_{\mathcal{T}_i}, \boldsymbol{\theta}, \alpha, m)}) \tag{7}$$

where $\mathcal{L}^q_{\mathcal{T}_i}$ is the loss computed on the query set. The outer-loop optimization does the update with the outer-loop learning rate β as:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^q_{\mathcal{T}_i}(f_{\boldsymbol{\theta}'_i}) \tag{8}$$

This involves computing second-order gradients, i.e., the backward pass works through the update

step in Equation 6, which is a computationally expensive process. Finn et al. (2017) propose a first-order approximation, called FOMAML, which computes the gradients with respect to $\boldsymbol{\theta}'_i$ rather than $\boldsymbol{\theta}$. The outer-loop optimization step thus reduces to:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_{\boldsymbol{\theta}'_i} \mathcal{L}^q_{\mathcal{T}_i}(f_{\boldsymbol{\theta}'_i}) \tag{9}$$

During meta-testing, new tasks are learned from the support sets and the performance is evaluated on the corresponding query sets.

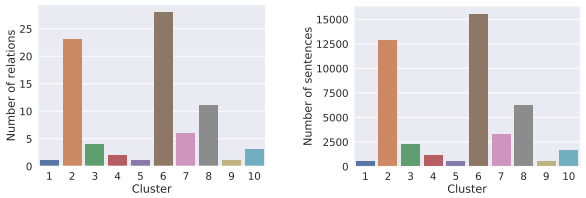
A.2 Dataset order for text classification

For text classification, the four different orderings of the datasets are:

1. Yelp → AGNews → DBpedia → Amazon → Yahoo
2. DBpedia → Yahoo → AGNews → Amazon → Yelp
3. Yelp → Yahoo → Amazon → DBpedia → AGNews
4. AGNews → Yelp → Amazon → Yahoo → DBpedia

A.3 Task distribution for relation extraction

In relation extraction, the size of each cluster is not balanced. Hence, each of the tasks vary in their size. In Figure 1 we plot the number of relations and the number of sentences in each cluster. Overall, there is a great imbalance with respect to the task size, with cluster 2 and 6 having a disproportionately larger size compared to the other clusters.



(a) Number of relations per cluster.

(b) Number of sentences per cluster.

Figure 1: Task distribution for relation extraction.

A.4 Illustration of the setup

Figure 2 provides an illustration of the structure of episodes and experience replay.

A.5 Motivation

Riemer et al. (2019) note that, given two sets of gradients for shared parameters $\boldsymbol{\theta}$, interference occurs when the dot product of gradients is negative, and transfer occurs when their dot product is positive. Additionally, they show that Reptile (Nichol et al., 2018) implicitly maximizes the dot product between gradients within an episode and, hence, when coupled with experience replay, it could facilitate continual learning.

Consider a first-order MAML setup that performs one step of SGD on each of the m batches in the support set during the inner-loop of an episode. Starting with parameters $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$, it results in a sequence of parameters $\boldsymbol{\theta}_1$, ..., $\boldsymbol{\theta}_m$ using the losses $\mathcal{L}^1$, ..., $\mathcal{L}^m$. The meta-gradient computed on the query set of the episode is:

$$g_{\text{FOMAML}} = \frac{\partial \mathcal{L}^q(\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}_m} \tag{10}$$

Using Taylor series approximation as in Nichol et al. (2018), the expected gradient under mini-batch sampling could be expressed as:

$$\mathbb{E}\left[g_{\text{FOMAML}}\right] = \mathbb{E}\left[\frac{\partial \mathcal{L}^q(\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}} - \frac{\alpha}{2}\frac{\partial}{\partial \boldsymbol{\theta}}\left(\sum_{j=1}^{m} \frac{\partial \mathcal{L}^j(\boldsymbol{\theta}_{j-1})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \mathcal{L}^q(\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}}\right)\right] + O(\alpha^2) \tag{11}$$

where α is the inner-loop learning rate. We provide a more detailed derivation in Appendix A.7. Outer-loop gradient descent with this gradient approximately solves the following optimization problem:

$$\min_{\theta} \mathbb{E}\left[\mathcal{L}^q(\boldsymbol{\theta}_m) - \frac{\alpha}{2}\left(\sum_{j=1}^{m} \frac{\partial \mathcal{L}^j(\boldsymbol{\theta}_{j-1})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \mathcal{L}^q(\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}}\right)\right] \tag{12}$$

This objective seeks to minimize the loss on the query set along with maximizing the dot product between the support and query set gradients. Thus, integrating previously seen examples into the query set in a first-order MAML framework could also potentially improve continual learning by minimizing interference and maximizing transfer.

A.6 Expression for replay frequency

In REPLAY and A-GEM, since gradient updates occur after seeing a batch of size b from the stream, the replay frequency R_F, i.e., the number of steps between the replay interval R_I, is simply given by

$$R_F = \left\lceil \frac{R_I}{b} \right\rceil \tag{13}$$

In meta-learning, learning occurs in episodes where the support set has m batches of size b each and a single batch as query set of the same size b. After encountering R_I examples, we would like the

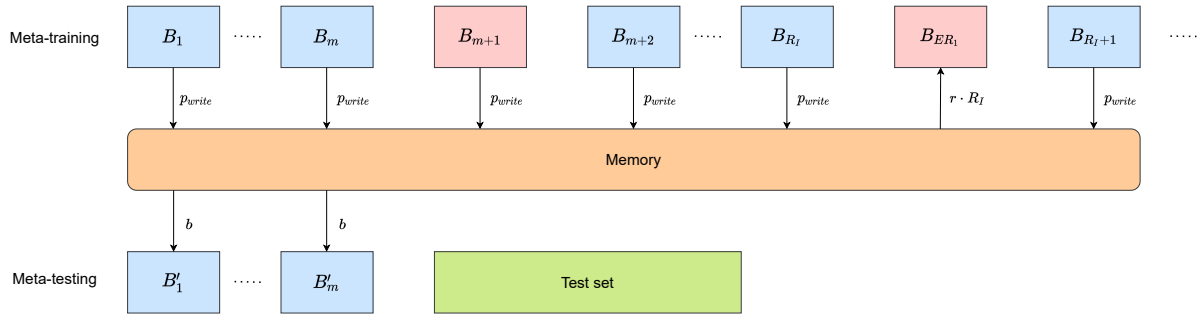


Figure 2: Illustration of meta-learning for lifelong learning. The m mini-batches that form the support set used for inner-loop optimization are shown in blue. Red boxes indicate query sets used in outer-loop optimization. After every R_I examples, the query set is obtained by sampling $r \cdot R_I$ examples from the memory, whereas other query sets are derived from the data stream. During meta-testing, m mini-batches are sampled from the memory for fine-tuning, followed by evaluation on the test set.

replay to be realized as a query set. If R_F is the episode at which replay occurs,

$$b\left[(R_F-1)(m+1)+m\right] = R_I$$
$$R_F - 1 = \frac{R_I/b - m}{m+1}$$
$$R_F = \left\lceil \frac{R_I/b+1}{m+1} \right\rceil \tag{14}$$

where we round it up to the nearest integer so that replay is not performed before R_I examples.

A.7 Derivation of gradients

Consider a first-order MAML setup that performs one step of SGD on each of the m batches in the support set during the inner-loop of an episode. Starting with parameters $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$, it results in a sequence of parameters $\boldsymbol{\theta}_1$, ..., $\boldsymbol{\theta}_m$ using the losses $\mathcal{L}^1$, ..., $\mathcal{L}^m$. The query set could be considered as the $(m+1)$-th batch that produces the meta-gradient for $\boldsymbol{\theta}$ using $\mathcal{L}^{(m+1)} = \mathcal{L}^q$. We introduce the following two notations to denote the gradient and the Hessian with respect to the initial parameters $\boldsymbol{\theta}$:

$$\bar{g}_i = \frac{\partial \mathcal{L}^i(\boldsymbol{\theta}_{i-1})}{\partial \boldsymbol{\theta}} \tag{15}$$
$$\bar{H}_i = \frac{\partial^2 \mathcal{L}^i(\boldsymbol{\theta}_{i-1})}{\partial \boldsymbol{\theta}^2} \tag{16}$$

Using Taylor series approximation, Nichol et al. (2018) show that the meta-gradient can be written as:

$$g_{\text{FOMAML}} = \frac{\partial \mathcal{L}^q(\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}_m}$$
$$= \bar{g}_{m+1} - \alpha \bar{H}_{m+1} \sum_{j=1}^{m} \bar{g}_j + O(\alpha^2)$$

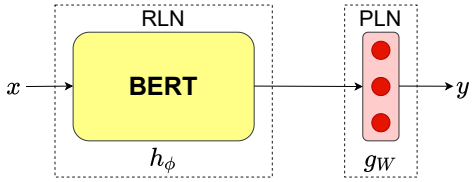


Figure 3: Architecture of OML.

Taking expectation under mini-batch sampling,

$$\mathbb{E}[g_{\text{FOMAML}}] = \mathbb{E}\left[\bar{g}_{m+1}\right] - \alpha \sum_{j=1}^{m} \mathbb{E}\left[\bar{H}_{m+1}\bar{g}_j\right] + O(\alpha^2)$$
$$= \mathbb{E}\left[\bar{g}_{m+1}\right] - \alpha \sum_{j=1}^{m} \mathbb{E}\left[\bar{H}_j\bar{g}_{m+1}\right] + O(\alpha^2)$$
(interchanging j and $m+1$)
$$= \mathbb{E}\left[\bar{g}_{m+1}\right] - \frac{\alpha}{2} \sum_{j=1}^{m} \mathbb{E}\left[\bar{H}_{m+1}\bar{g}_j + \bar{H}_j\bar{g}_{m+1}\right]$$
$+ O(\alpha^2)$ (averaging the last two equations)
$$= \mathbb{E}\left[\bar{g}_{m+1}\right] - \frac{\alpha}{2}\frac{\partial}{\partial \boldsymbol{\theta}} \sum_{j=1}^{m} \mathbb{E}\left[\bar{g}_j \cdot \bar{g}_{m+1}\right]$$
$+ O(\alpha^2)$

Re-writing based on Equation 15 and 16 gives:

$$\mathbb{E}[g_{\text{FOMAML}}] = \mathbb{E}\left[\frac{\partial \mathcal{L}^q(\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}} - \frac{\alpha}{2}\frac{\partial}{\partial \boldsymbol{\theta}}\left(\sum_{j=1}^{m} \frac{\partial \mathcal{L}^j(\boldsymbol{\theta}_{j-1})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \mathcal{L}^q(\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}}\right)\right] + O(\alpha^2) \tag{17}$$

A.8 Model architecture

Figure 3 and 4 depict the architectures of OML and ANML respectively.

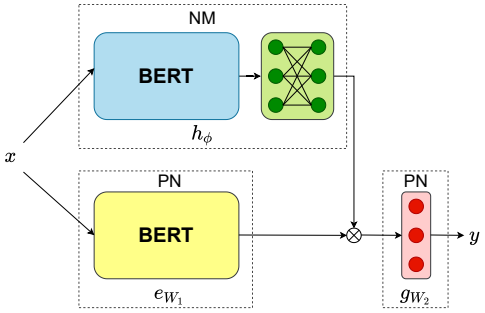


Figure 4: Architecture of ANML.

Algorithm 2: Meta-testing

Input: Trained model parameters $\boldsymbol{\theta} = \boldsymbol{\phi} \cup \boldsymbol{W}$, support set buffer size m, memory $\mathcal{M}$, batch size b, inner-loop learning rate α, test set T

Output: Predictions on the test set

$\mathcal{S} \leftarrow \text{sample}(\mathcal{M}, m \cdot b)$

$\mathcal{Q} \leftarrow T$

$\boldsymbol{W}' = \text{SGD}(\mathcal{L}, \boldsymbol{\phi}, \boldsymbol{W}, \mathcal{S}, \alpha)$

$\text{predict}(\mathcal{Q}, \boldsymbol{\phi}, \boldsymbol{W}')$

A.9 Pseudo code

Algorithms 1 and 2 outline the meta-training and meta-testing procedure respectively that is common to both OML-ER and ANML-ER.

Algorithm 1: Meta-training

Input: Initial model parameters $\boldsymbol{\theta} = \boldsymbol{\phi} \cup \boldsymbol{W}$, replay interval R_I, replay frequency R_F, replay rate r, support set buffer size m, memory $\mathcal{M}$, write probability p_{write}, inner-loop learning rate α, outer-loop learning rate β

Output: Trained model parameters $\boldsymbol{\theta}$, updated memory $\mathcal{M}$

for $i = 1, 2, \ldots$ **do**

  $\mathcal{S}_i \leftarrow m$ batches from the stream

  if $i = R_F$ **then**

    $\mathcal{Q}_i \leftarrow \text{sample}(\mathcal{M}, \lfloor r \cdot R_I \rfloor)$

  end

  else

    $\mathcal{Q}_i \leftarrow$ next batch from the stream

    $\text{write}(\mathcal{M}, \mathcal{Q}_i, p_{write})$

  end

  $\text{write}(\mathcal{M}, \mathcal{S}_i, p_{write})$

  $\boldsymbol{W}'_i = \text{SGD}(\mathcal{L}_i, \boldsymbol{\phi}, \boldsymbol{W}, \mathcal{S}_i, \alpha)$

  $J(\boldsymbol{\theta}) = \mathcal{L}_i\left(\boldsymbol{\phi}, \boldsymbol{W}'_i, \mathcal{Q}_i\right)$

  $\boldsymbol{\theta} \leftarrow \text{Adam}(J(\boldsymbol{\theta}), \beta)$

end

A.10 Implementation details

For text classification, we take $5,000$ examples from each of the datasets as the validation set and $115,000$ examples from each of the datasets as the training set. The number of training examples matches that of d'Autume et al. (2019). On the other hand, for relation extraction, we take a subset of $4,800$ examples from the training set as the validation set for hyperparameter tuning. With the best hyperparameters, we re-train on all $44,800$ examples to match the number of examples used in Wang et al. (2019).

The only hyperparameters we tune are the learning rate (for SEQ, A-GEM and REPLAY), the inner and meta learning rates, and the support set buffer size m (for OML-ER and ANML-ER). The other hyperparameters are fixed to appropriate values. We performed tuning over the following values:

- Learning rate: $5e{-}4$, $1e{-}5$, $3e{-}5$, $5e{-}5$
- Inner learning rate: $5e{-}2$, $1e{-}3$, $3e{-}3$, $5e{-}3$
- Meta learning rate: $5e{-}4$, $1e{-}5$, $3e{-}5$, $5e{-}5$
- m: 3, 5, 7, 9

In Table 6, we summarize all the hyperparameters for text classification and relation extraction. We use the random seeds 42 – 44 for the three independent runs. All models were trained on a system with a single Nvidia Titan RTX GPU and 45 GB memory.

A.11 Additional results

Table 7 shows the test set accuracy on the text classification benchmark per order of the dataset as well as the macro average.

A.12 Frequency of constraint violations

A-GEM solves a constrained optimization problem such that the dot product between the gradients from the current batch and a randomly drawn batch from the memory is greater than or equal to zero. We check constraint satisfaction by treating the model parameters as a single vector. To analyze the poor performance of A-GEM on our setup, we plot the average number of constraint violations across the four orders that occur per task in text classification in Figure 5. Note that the total number of optimizer steps per task is 7187 and replay

Model	Learning rate	Inner loop learning rate	Meta learning rate	Support set buffer size	Batch size	Maximum sequence length
SEQ	3e−5	—	—	—	16	448
A-GEM	3e−5	—	—	—	16	448
REPLAY	3e−5	—	—	—	16	448
MTL (2 epochs)	3e−5	—	—	—	16	448
OML-ER	—	1e−3	1e−5	5	16	448
ANML-ER	—	3e−3	1e−5	5	16	300
SEQ	3e−5	—	—	—	4	—
A-GEM	3e−5	—	—	—	4	—
REPLAY	3e−5	—	—	—	4	—
MTL (3 epochs)	1e−5	—	—	—	4	—
OML-ER	—	1e−3	3e−5	5	4	—
ANML-ER	—	1e−3	3e−5	5	4	—

Table 6: Hyperparameters for text classification (top) and relation extraction (bottom).

Method	Accuracy				
	Order 1	Order 2	Order 3	Order 4	Average
MbPA++ (d'Autume et al., 2019)	70.8	70.9	70.2	70.7	70.6
MbPA++ (Sun et al., 2020)	74.1	74.9	73.1	74.9	74.2
LAMOL (Sun et al., 2020)	76.7	77.2	76.1	76.1	76.5
Meta-MbPA (Wang et al., 2020)	77.9	76.7	77.3	77.6	77.3
SEQ	16.7 ± 0.7	25.0 ± 0.5	19.5 ± 0.4	22.1 ± 0.5	20.8 ± 0.5
A-GEM	16.6 ± 0.9	25.9 ± 1.1	21.6 ± 0.8	23.5 ± 1.0	21.9 ± 0.3
REPLAY	69.5 ± 1.0	66.2 ± 2.0	65.2 ± 2.3	68.3 ± 2.2	67.3 ± 0.7
OML-ER	75.4 ± 0.3	**76.5 ± 0.2**	75.4 ± 0.5	75.4 ± 0.8	75.7 ± 0.4
ANML-ER	**75.6 ± 0.4**	75.8 ± 0.1	**75.5 ± 0.3**	**75.7 ± 0.3**	**75.7 ± 0.1**
MTL	—	—	—	—	79.4 ± 0.2

Table 7: Test set accuracy on text classification. The last column is the macro average across the four orders.

occurs about 11 times for each. When fine-tuning the whole of BERT, we have relatively few violations, meaning that no gradient correction is done most of the time. This perhaps relates to the finding by Merchant et al. (2020) that fine-tuning BERT primarily affects the top layers and does not lead to catastrophic forgetting of linguistic phenomena in the deeper layers. We see that the number of violations increase when we only fine-tune the top 2 layers of BERT. Yet, it was insufficient to reach the performance of a simple replay method.

A.13 ANML visualization

The original OML and ANML models were shown to produce sparse representations with CNN encoders for images (Javed and White, 2019; Beaulieu et al., 2020). Sparse representations alleviate forgetting since only a few neurons are active for a given input. We visualize the representations from BERT before and after neuromodulation, along with the neuromodulatory signal, in our ANML-ER model in Figure 6. Clearly, none of the representations are sparse. Moreover, most of the neuromodulatory signal is composed of ones,

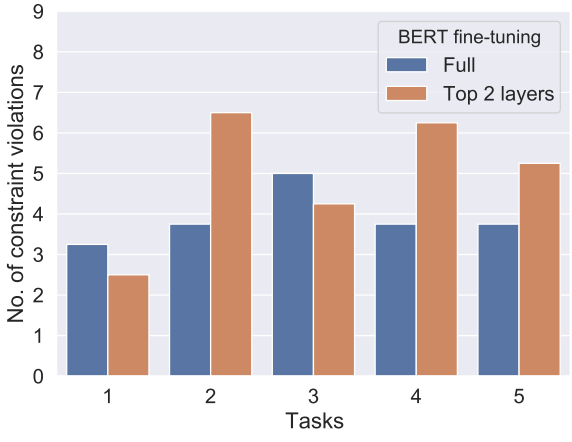


Figure 5: Average number of constraint violations per task in text classification.

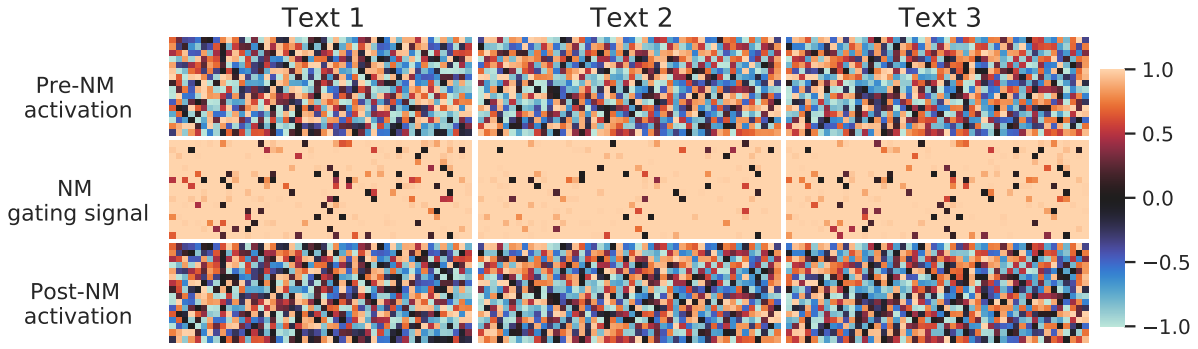


Figure 6: Visualization of the neuromodulatory signal (middle row) and the representation from BERT before (top row) and after (bottom row) neuromodulation for three randomly chosen texts from the AGNews dataset. We obtain the plots by reshaping the 768-dimensional representation into 48×16.

further confirming our hypothesis that the neuromodulator does not play a significant role here. The lack of sparsity was also observed in OML-ER. Perhaps, a more sophisticated neuromodulatory mechanism is required to induce sparsity in pre-trained transformer-based language models.