Prototype-Guided Memory Replay for Continual Learning

Stella Ho, Ming Liu, Lan Du, Longxiang Gao, Senior Member, IEEE, and Yong Xiang, Senior Member, IEEE

Abstract—Continual learning (CL) is a machine learning paradigm that accumulates knowledge while learning sequentially. The main challenge in CL is catastrophic forgetting of previously seen tasks, which occurs due to shifts in the probability distribution. To retain knowledge, existing CL models often save some past examples and revisit them while learning new tasks. As a result, the size of saved samples dramatically increases as more samples are seen. To address this issue, we introduce an efficient CL method by storing only a few samples to achieve good performance. Specifically, we propose a dynamic prototype-guided memory replay (PMR) module, where synthetic prototypes serve as knowledge representations and guide the sample selection for memory replay. This module is integrated into an online meta-learning (OML) model for efficient knowledge transfer. We conduct extensive experiments on the CL benchmark text classification datasets and examine the effect of training set order on the performance of CL models. The experimental results demonstrate the superiority of our approach in terms of accuracy and efficiency.

Index Terms—Class-incremental learning (CIL), continual learning (CL), online meta-learning (OML), prototypical network, text classification.

NOMENCLATURE

- $k$: Task index.
- $l$: Class index.
- $i$: Episode index.
- $N$: Number of classes.
- $f_0$: Model with parameters $\theta$.
- $h_{\text{proto}}$: Prototypical network with parameters $\phi_{\text{proto}}$.
- $\mathbf{S}_{\text{pred}}$: Classifier with parameters $\phi_{\text{pred}}$.
- $D_{\text{train}}$: Training set.
- $\mathcal{M}$: Memory set.
- $S$: Support set for meta-learning.
- $Q$: Query set for meta-learning.
- $\mathcal{L}_{\text{CE}}$: Cross entropy loss.
- $\mathcal{L}_P$: Prototypical loss for a class.
- $J_P(\cdot)$: Prototypical loss for a training episode.
- $N_S$: Number of support examples per class.
- $N_Q$: Number of query examples per class.
- $N_K$: Number of saved examples per class.
- $c$: Prototype set.
- $m$: Optimization step.
- $b$: Batch size.
- $R_F$: Replay frequency.
- $d(\cdot)$: Distance function.
- $\alpha$: Inner loop learning rate.
- $\beta$: Outer loop learning rate.

I. INTRODUCTION

CURRENT dominant machine learning paradigms often learn from identically and independently distributed (i.i.d.) data. It often requires a large amount of training samples to ensure models’ performance. In real-world scenarios, the availability of data is limited. The high-quality training data may only become accessible over time. Conventional machine learning models always learn from scratch to update knowledge. Yet, some past examples may become inaccessible due to privacy reasons, causing the models to fail to retain previous knowledge. Hence, current dominant learning paradigms are incapable of learning continuously and effectively.

Continual learning (CL) aims to address these problems. Intuitively, it mimics human learning progress by accumulating knowledge and learning built upon it. The main challenge in CL is catastrophic forgetting [1]. It manifests as evident degradation in the model’s performance on preceding tasks. Catastrophic forgetting is also known as the stability–plasticity dilemma. CL aims to guarantee the stability of handling various learned tasks while showing plasticity in a novel domain.

The existing CL models can be categorized into three main-stream methods, i.e., regularization-based methods [2], [3], [4], architecture-based methods [5], [6], [7], [8], and replay-based methods [9], [10], [11], [12], [13]. The regularization-based method typically limits the updates of nontrivial parameters or weights by adding constraints. Due to the complexity of deep learning models, a small change in parameter space could have unexpected effects on the model’s performance [9], [14]. The architecture-based method dynamically expands the
model architecture to learn new tasks. It preserves previously fine-tuned parameters and introduces task-specific parameters for new tasks. However, the size of the parameters grows as the number of seen tasks increases. Most existing CL models are the replay-based methods. The replay-based method revisits past data or generates pseudo past data to consolidate knowledge. However, most replay-based models neglect the memory constraint in the CL setup. The size of the saved samples grows aggressively as more tasks are seen. Hence, it is imperative to efficiently save and use samples for memory replay.

This article introduces a prototype-guided memory replay (PMR) module, which dynamically updates knowledge. We use a prototypical network as a feature extractor. It generates prototypes as knowledge representations. We use prototypes to guide the sample selection for memory replay. We also leverage meta-learning to enable fast adaptation and knowledge transfer. In the experiments, we use datasets from [15], popularized by de Masson d’Autume et al. [16] in lifelong language learning. This collection of datasets consists of text classification datasets from diverse domains. The replay rate of our model is set to a relatively low value, not exceeding 0.8%. Extensive experimental results demonstrate the superiority of the proposed method in terms of accuracy and efficiency. Particularly, we reduce the memory budget by keeping no more than 0.2% of seen samples.

Our main contributions are threefold: 1) we propose a CL model, called PMR, which combines an online meta-learning (OML) model with a prototypical network and enables efficient knowledge consolidation through revisiting a few past examples; 2) we devise a PMR module, where prototypes are dynamically updated and guide sample selection, providing sample efficiency in memory; and 3) in experiments, we show that the model’s performance is sensitive to the order of training sets when data distributions are significantly different.

The remainder of this article is organized as follows. We present the related work in Section II. We then introduce the problem formulation of CL in Section III. Section IV depicts the proposed approach with an illustration of the model and detailed algorithms. In Section V, extensive experimental outcomes are thoroughly discussed and analyzed. Section VI concludes this article.

II. RELATED WORK
A. Replay-Based Methods

The replay-based method often saves previous training data or generates pseudo samples. It relieves catastrophic forgetting by revisiting them while learning new tasks, known as memory replay or experience rehearsal. Many CL models use random selection to select samples for replay, which neglect the memory constraint in CL and fail to consider memory efficiency. To retain knowledge, gradient episodic memory (GEM) [17] and A(verage)-GEM [11] constrain gradient projection by revisiting past examples, in which models randomly save samples and replay samples at every training step. Recent replay-based methods use episodic memory replay [16] with a 1% replay rate to prevent overfitting to past examples. Memory-based parameter adaptation (MbPA) ++ [16] retrieves K-nearest neighbors (KNNs) for local adaptation in its inference process, which is expensive. Holla [18] introduces an experience replay mechanism to harness the replay sparseness in terms of time and size. References [12], [19], and [20] are generative replay-based models, which generate pseudo samples to bypass storage of real data.

1) Sample Selection Schemes: The widely used sample selection criteria are random selection and KNN. Most replay-based methods randomly select data from a memory buffer or randomly save past training data. For a task-specific learning process, some approaches [16], [21], [22] apply K-center method to choose K data points according to their distances to centroid or other geometric properties. Both the random selection methods and K-center methods use heuristics to update memory. Embedding aligned episodic memory replay (EA-EMR) [9] and information disentanglement-based regularization (IDBR) [23] select informative samples by referencing the centroid of the cluster via K-means. Incremental classifier and representation learning (iCaRL) method [24] chooses samples that are nearest to the mean of the distribution. Lee et al. [25] use short-term memory as a memory buffer to collect sufficient newly seen data, thereby preventing biasing toward the current learning task. Wang et al. [10] apply two popular paradigms in active learning into memory selection rules, i.e., diversity-based methods [26] and uncertainty-based methods [27], [28]. Nevertheless, random selection is considered as the most efficient strategy compared with these selection rules.

B. Continual Learning in NLP

Typically, language models are deep learning models, while regularization-based methods operate on the parameter spaces. However, due to the complexity of deep learning models, such an approach can be challenging [29] and is not considered as a proper method for constraining model behavior. To address natural language processing (NLP) tasks, most CL approaches favor replay-based methods and carefully impose regularization in their embedding spaces [9], [23]. For instance, EA-EMR uses an embedding alignment model to reduce distortion in the embedding space. IDBR [23] uses information disentanglement to differentiate generic and task-specific representations in hidden spaces, allowing only the task-specific representation to undergo a large degree of updates during training. Sequence-to-sequence CL (S2S-CL) [30] leverages compositionality [31] by dividing the input into semantic and syntactic representations, with the syntactic representation remaining frozen during training.

C. Meta-Learning in Continual Learning

Recently, meta-learning has been introduced into CL, due to its ability to facilitate fast adaptation and knowledge transfer. Several works [10], [18] use model-agnostic meta-learning (MAML) to optimize initial parameters, enabling fast adaptation to various domains. MetaMbPA [10] combines local adaptation with episodic memory replay and uses MAML to find a better initialized state for local adaptation. OML-ER [18] and neuromodulated meta-learning (ANML)-ER [18] use an OML model and an ANML framework, respectively, for fast adaptation, augmented with
episodic experience replay. In addition, some CL models use Reptile [32] as their meta-learning frameworks. Meta-experience replay (MER) [33] regularizes the objective of experience replay via a modified Reptile [32] algorithm and memory replay module. Meta-learning for lifelong relation extraction (MLLRE) [21] also adopts Reptile to meta-update parameters via augmented training set. Furthermore, in the field of computer vision, meta-consolidation for continual learning (MERLIN) method [34] uses preceding task-specific priors from meta distribution to replay previous parameters and consolidate model. Memory efficient online learning (MOML) method [35] introduces quadratic penalty to debias and regularize the loss of a meta model, such that it can bypass the need to recall prior seen instances.

**D. Lifelong Text Classification Settings**

CL has two settings in text classification, i.e., class-incremental learning (CIL) and task incremental learning (TIL). In CIL, task identity information is not provided in inference, and it has a single classifier for all the classes. While in TIL, task identity information is provided in both training and inference, and a CL model assigns one classifier for each task. Recent works mainly operate in a task-free setting [36] where neither training nor test examples contain dataset identity.

**III. PROBLEM FORMULATION**

We consider a basic CL setup where model makes a single pass over a stream of training examples. Assume the training data stream consists of $K$ tasks, $(\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(k)})$, in a specific order. Each task $\mathcal{T}^{(k)}$ is a supervised learning task with a ground-truth label set $D_{\text{train}}^{(k)} = \{(x_j^{(k)}, y_j^{(k)})\}_{j=1}^{N^{(k)}}$. The objective for a task $\mathcal{T}^{(k)}$ is to train a continual learner $f$ that converges to the probability distribution of $\mathcal{T}^{(k)}$, while still performing well on previous tasks $\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(k-1)}\}$ without using the past training sets. The overall objective is to minimize the average expected risk of all $K$ tasks seen so far, i.e., $(1/K)\sum_{k=1}^{K} \mathbb{E}_{x,y \sim P(\mathcal{T}^{(k)})}[L(f_\theta(x), y)]$, where $K$ denotes the total number of seen tasks and $\theta$ is the parameters of $f$. A replay-based method allows the model $f$ to save a certain amount of training samples from previous tasks. In our CL setup, we constrain the memory size to a constant value of $B$, with the memory storing a maximum of $B$ samples from previous tasks.

**A. Class-Incremental Learning**

We address text classification tasks in a CIL scenario, where task identity information is not given in inference. Given an encoder $h_{\phi_{\text{proto}}}: \mathcal{X} \rightarrow \mathbb{R}^d$ and a classifier $g_{\phi_{\text{pred}}}: \mathbb{R}^d \rightarrow \mathbb{R}^N$, where $d$ is the dimension of the hidden representations and $N$ is the number of classes, we consider a model $f_\theta = g_{\phi_{\text{pred}}}(h_{\phi_{\text{proto}}}(x))$. The cross entropy loss for a task $\mathcal{T}^{(k)}$ is

$$\mathcal{L}_{\text{CE}}(x, y; \theta^{(k)}) = -\sum_{j=1}^{N^{(k)}} \sum_{i=1}^{N} y_{ji} \log(\sigma(f_{\theta^{(k)}}(x)_i))) \quad (1)$$

where $N^{(k)}$ is the number of classes in $\mathcal{T}^{(k)}$, $\sigma$ is the activation function (e.g., sigmoid or softmax), and $(x, y) \in D_{\text{train}}^{(k)}$.

Alternatively, the cross entropy loss while learning a task $\mathcal{T}^{(k)}$ can be described as the loss for all the seen classes. For a replay-based methods, the cross entropy loss generally is

$$\mathcal{L}_{\text{CE}}^*(x, y; \theta^{(k)}) = -\sum_{j=1}^{|D_{\text{train}}^{(k)}|} \sum_{i=1}^{N} y_{ji} \log(\sigma(f_{\theta^{(k)}}(x)_i))) \quad (2)$$

where $\mathcal{M}$ is the memory set that contains past examples and $(x, y) \in D_{\text{train}}^{(k)} \cup \mathcal{M}$. In this article, we mainly use the cross entropy loss shown in (1). More details are shown in Section IV-A.

**IV. CONTINUOUS LEARNING WITH PROTOTYPE-GUIDED MEMORY REPLAY**

Meta-learning is a widely used machine learning paradigm for fast adaptation and efficient knowledge transfer [18]. We combine a meta-learning framework with our devised PMR module to formulate a CL approach, namely, PMR.

**A. Meta Continual Learning**

Recently, some CL models [10], [18] have used meta-learning to learn an optimized initial state of the parameters, allowing the model to perform well on all seen tasks with only a few gradient updates. The meta-training process involves a two-level optimization process. In the inner loop, the model performs task-specific fine-tuning on the support set $S$. In the outer loop, the model performs meta-updates for meta-objectives using the query set $Q$. Given a model $f$ with parameter set $\theta$, the inner loop optimization is a $m$-step gradient-based update on the support set $S$. We denote the fine-tuned parameters on the support set $S$ as $\theta'$. The outer loop algorithm aims to generalize well with the fine-tuned parameters $\theta'$. However, it involves second-order gradient computation with respect to $\theta$, which is computationally expensive. First-order MAML (FOMAML) [37] provides a solution to this problem using a first-order approximation, where the gradients are computed with respect to the fine-tuned parameters $\theta'$. Therefore, we choose FOMAML as our meta-learning framework for its simplicity.

1) Architecture: The proposed model $f_\theta$ consists of a representation learning network (RLN) and a prediction network (PN). We add a single hidden layer feedforward neural network on top of an encoder to build a prototypical network as RLN. We denote RLN as $h_{\phi_{\text{proto}}}$ with learnable parameters $\phi_{\text{proto}}$. We use a single linear layer followed by a softmax as PN. We denote PN as $g_{\phi_{\text{pred}}}$ with learnable parameters $\phi_{\text{pred}}$. The model is defined as $f_\theta(x) = g_{\phi_{\text{pred}}}(h_{\phi_{\text{proto}}}(x))$.

2) Online Meta-Learning: In each episode $i$, we instantaneously sample $m$ batches of examples from the data stream $D_{\text{train}}$ as the support set $S$. The inner loop loss $\mathcal{L}_{\text{inner}}$ consists of a cross entropy loss $\mathcal{L}_{\text{CE}}$ and a prototypical loss $J_P$ as

$$\mathcal{L}_{\text{inner}}(S; \theta) = \mathcal{L}_{\text{CE}}(S; \theta) + J_P(\phi_{\text{proto}}) = \mathcal{L}_{\text{CE}}(S; \phi_{\text{proto}}) \cup \mathcal{L}_{\text{CE}}(S; \theta) + J_P(\phi_{\text{proto}}). \quad (3)$$

The computation of $J_P(\phi_{\text{proto}})$ is shown in (9). The inner loop algorithm only fine-tunes the PN model $g_{\phi_{\text{pred}}}$ via stochastic gradient descent (SGD) with learning rate $\alpha$ as $\phi'_{\text{pred}} = \phi_{\text{pred}} - \alpha \nabla_{\theta} \mathcal{L}_{\text{inner}}(S; \theta)$. In the outer loop, both the
parameters \( \phi_{\text{proto}} \) and \( \phi_{\text{pred}} \) are meta-learned using the query set \( Q \). The meta loss is defined as

\[
L_{\text{outer}}(Q; \theta') = L_{\text{CE}}(Q; \theta'),
\]

where \( R_F \) is the replay frequency and \( \theta' = \phi_{\text{proto}} \cup \phi_{\text{pred}} \). \( Q \) is a batch of examples from the data stream \( D_{\text{train}} \). The optimizer of the outer loop algorithm is the Adam optimizer [38] with a learning rate \( \beta \). The meta-training and inference process are described in Algorithms 1 and 2, respectively.

3) Task-Specific Classification Learning: Both the inner and outer loop algorithms apply cross-entropy loss for task-specific learning. The cross entropy loss for the inner loop is

\[
L_{\text{CE}}(S; \theta) = L_{\text{CE}}(S; \phi_{\text{proto}} \cup \phi_{\text{pred}})
\]

\[
= -\sum_{j=1}^{\left| S \right|} \sum_{i=1}^{N^{(k)}} y_{ji} \log(\sigma(g_{\phi_{\text{proto}}}(h_{\phi_{\text{pred}}}(x_{ji}))))
\]

where \( N^{(k)} \) is the number of classes in the current task \( T^{(k)} \).

The cross entropy loss for the outer loop, when not performing experience replay, is

\[
L_{\text{CE}}(Q; \theta') = L_{\text{CE}}(Q; \phi_{\text{proto}} \cup \phi_{\text{pred}})
\]

\[
= -\sum_{j=1}^{\left| Q \right|} \sum_{i=1}^{N^{(k)}} y_{ji} \log(\sigma(g_{\phi_{\text{pred}}}(h_{\phi_{\text{proto}}}(x_{ji}))))
\]

4) Experience Replay: When \( i = R_F \), we perform experience replay by retrieving examples from the memory \( \mathcal{M} \) and using them as the query set \( Q \). As a result, the meta-objective for experience replay is formulated as

\[
L_{\text{outer}}(\mathcal{M}; \theta') = L_{\text{CE}}(\mathcal{M}; \phi_{\text{proto}} \cup \phi_{\text{pred}})
\]

\[
= -\sum_{j=1}^{\left| \mathcal{M} \right|} \sum_{i=1}^{N} y_{ji} \log(\sigma(g_{\phi_{\text{pred}}}(h_{\phi_{\text{proto}}}(x_{ji}))))
\]

where \( N \) is the number of classes in \( \mathcal{M} \). We define this loss as the experience replay loss. Minimizing the experience replay loss allows \( f_{\theta'} \) to converge to the probability distribution of all seen tasks. As for replay frequency, we activate memory retrieval from \( \mathcal{M} \) every 50 epochs, which is lower than 1% replay rate. More details are shown in Section V-C.

B. Prototype-Guided Memory Replay Module

We aim to use a limited amount of seen instances to consolidate knowledge. A prototypical network can characterize hidden representations through prototypes, even when only a few examples are available. To achieve sample efficiency, we propose a replay memory module in which prototypes participate in the write mechanisms, as depicted in Fig. 1.

1) Prototypical Network: While learning a task \( T^{(k)} \), our model also learns prototypes that represent the classes in that task. We propose a PMR module that selectively writes examples from the query set \( Q \) into memory \( \mathcal{M} \), using prototypical knowledge. The prototypical network \( h_{\phi_{\text{proto}}} \) is meta-updated with learnable parameters \( \phi_{\text{proto}} \) and dynamically produces or corrects prototypes. For a class \( l \), the model learns a \( d \)-dimensional feature vector \( c_l \) as the prototype. The prototype \( c_l \) is the mean vector of \( N_S \) embedded support examples labeled as \( l \), and it is computed as

\[
c_l \leftarrow \frac{1}{|S_l|} \sum_{(x_{ji}, y_{ji}) \in S_l} h_{\phi_{\text{proto}}}(x_{ji}).
\]

The query set for learning a class \( l \) is denoted as \( Q_l \). Note that \( S_l \cup Q_l \subseteq S \). For a task \( T^{(k)} \), the prototypical loss [39] of a class \( l \) is formulated as

\[
L_{P}(Q_l; \phi_{\text{proto}}) = \frac{1}{N_Q^{(k)}} \sum_{l} \left[ d(h_{\phi_{\text{proto}}}(x), c_l) + \log \sum_{l'} \exp(-d(h_{\phi_{\text{proto}}}(x), c_{l'})) \right]
\]

where \( N^{(k)} \) denotes the number of classes in \( T^{(k)} \), \( N_Q \) denotes the number of query examples for each class, and \( d(\cdot) \) denotes the distance function. In this article, we leverage Euclidean distance.
2) Dynamic Memory: As a reference for sample selection, we constantly update prototypes with a meta-learnable prototypical network \(h_{\phi_{\text{prot}}}\). The samples in memory \(\mathcal{M}\) are dynamically updated with the prototypes. Specifically, we use the KNN method to obtain \(N_K\) examples of a class \(l\) based on their distance to the prototype \(c_l\). In this article, we select five instances per class from both the query set \(\mathcal{Q}\) and the memory set \(\mathcal{M}\). The previously stored examples are then replaced with the newly selected examples, updating memory set \(\mathcal{M}\). In this approach, the memory size is increased linearly with the number of seen classes, not with the number of seen examples, thereby reducing memory budgets.

**Algorithm 1 Meta-Training**

**Input:** Initial parameters \(\theta = \phi_{\text{proto}} \cup \phi_{\text{pred}},\) training set \(D_{\text{train}},\) support set size \(m,\) memory buffer \(\mathcal{M},\) inner loop learning rate \(\alpha,\) outer loop learning rate \(\beta,\) and number of saved examples per class \(N_K.\)

**Output:** Trained parameters \(\theta\) and memory set \(\mathcal{M}\)

```
for i = 1, 2, \ldots do
    S \leftarrow m \text{ batches of examples } D_i^{\text{train}} \text{ from the stream.}
    \text{[Prototypical Network]}
    J_{\phi_{\text{proto}}} \leftarrow 0
    \text{for class } l \in S \text{ do}
        \text{Select } N_S \text{ examples of class } l \text{ from } S \text{ as } S_l.
        \text{Select } N_Q \text{ examples of class } l \text{ from } S \setminus S_l \text{ as } Q_l.
        \text{Compute prototype } c_l \text{ as } 8.
        \text{Update } c_l \text{ in prototype set } c.
        \text{Compute prototypical loss } \mathcal{L}_{\phi_{\text{proto}}}(Q_l; \phi_{\text{proto}}) \text{ as } 9.
        J_{\phi_{\text{proto}}} \leftarrow J_{\phi_{\text{proto}}} + \mathcal{L}_{\phi_{\text{proto}}}(Q_l; \phi_{\text{proto}})
    end
    if i = R_F \text{ then}
        Q \leftarrow \mathcal{M}, \text{ a set of all examples from memory.}
    else
        Q \leftarrow \text{next batch of examples } D_i^{\text{train}} \text{ from the stream.}
        \text{[Samples Selection and Write Mechanism]}
        \text{for class } l \in \text{ prototype set } c \text{ do}
            \text{Select } N_K \text{ nearest examples to } c_l \text{ from } Q \cup \mathcal{M}.
            \text{Update examples of class } l \text{ in } \mathcal{M}.
        end
    end
    \text{[Inner Loop]}
    \text{Perform SGD on } \phi_{\text{pred}} \text{ to minimize } 3 \text{ as}
    \phi_{\text{pred}} = \phi_{\text{pred}} - \alpha \nabla \mathcal{L}_{\phi_{\text{inner}}}(S; \theta).
    \text{[Outer Loop]}
    \text{Perform Adam update on } \theta \text{ to minimize } 4 \text{ as}
    \theta \leftarrow \theta - \beta \nabla \mathcal{L}_{\phi_{\text{outer}}}(Q; \theta'), \text{ where}
    \theta' = \phi_{\text{proto}} \cup \phi_{\text{pred}}.
    \text{if all training data are seen then}
        \text{Stop Training}
    end
end
```
10978 IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 35, NO. 8, AUGUST 2024

1) A-GEM [11] replays past samples at every training step by default. We follow the settings of prior work [16], [18], in which AGEM randomly revisits examples from memory with a 1% replay rate.

2) Replay is a simple sequential learning baseline with a 1% replay rate. It uses the widely used write and read mechanisms, i.e., it writes all and randomly reads.

3) OML-ER [18] uses the same OML [40] framework as the proposed model. Its default setting involves writing all seen data and randomly reading with a 1% replay rate. We adapt it to our setting by limiting the size of its memory buffer for past data.

4) ANML-ER [18] uses the ANML [41] framework. It has the same write and read mechanisms as OML-ER. We reduce the memory limit of ANML-ER by allowing it to save 25 samples per class in each episode.

5) PMR is our proposed model, but with a higher replay rate, i.e., 1% replay rate.

C. Setup

In our experiments, we limit the memory buffer size to store only a few past examples. Specifically, we allow the buffer to store up to five samples per class \((N_k = 5)\), with a total of 45 instances for the datasets, Yelp, AGNews, and Amazon (i.e., \(B = 0.01\%\) of seen data). The write rate, \(p_{\text{write}}\), is set to 1. We use the pretrained ALBERT-Base-v2 from Hugging Face Transformers [42] as our example encoder, with a truncated input sequence length of 200. For the prototypical network, we use ReLU(B7) as the activation function with a dropout rate 0.2. The inner loop optimizer for OML-ER, ANML-ER, and PMR is SGD with a learning rate of \(\alpha = 1e^{-3}\), while the outer loop optimizer is Adam with a learning rate of \(\beta = 3e^{-5}\). A-GEM, Replay, OML-ER, and ANML-ER use a random sampler, which randomly samples examples with a batch size of \(b = 25\). PMR samples examples uniformly at random without replacement, with batch sizes that depend on the number of classes in the training sets, e.g., \(b = 20\) for AGNews and \(b = 25\) for Yelp and Amazon. For prototypical network, we have \(N_S = 3\) and \(N_Q = 2\). The support set size in each episode is \(m = 5\). All the baseline models use a 1% experience replay rate, while the replay frequency of PMR is every 50 epochs, resulting in different replay rates (replay rate = \((|M|/b \cdot (m + 1) \cdot R_{F} + b \cdot m))\) shown in Table II.

All the models are run on a Linux platform with eight Nvidia Tesla A100 GPUs and 40 GB of RAM. All the experiments are performed using PyTorch.\(^1\)

D. Results

We record the average of the three best results from five trials and report the average accuracy across all six different sequences. All the models are trained with a fixed number of epochs without multiple iterations. As shown in Table III, the PMR models, namely, PMR and PMR\(_{1\%}\), obtain the best result in terms of average accuracy by keeping only 0.01% of the seen data for memory replay. In particular, the PMR models achieve the highest accuracy in four sequences. The result of PMR and PMR\(_{1\%}\) also indicates that replay rate affects model’s performance. The effect of replay rate is discussed in Section V-E6. Given a less than 1% replay rate, PMR outperforms all the baselines by at least 1.89% in terms of average accuracy, making it the most efficient method compared with the baselines. Only OML-ER shows a competitive result, indicating the superiority of the OML framework in alleviating catastrophic forgetting. Furthermore, the average accuracy of A-GEM is around 2.18% lower than Replay, aligning with the results reported in literature [10], [18]. This result suggests that using past examples as gradient constraints may not be as effective as using them as part of the training data, especially for NLP tasks. Surprisingly, ANML-ER performs inadequately, possibly due to its inapplicability of the strict memory constraint. It is evident that the performance of all the models is sensitive to the order of training sets. All the models perform poorly in sequences Y → A → AGN and A → Y → AGN. The reasons behind are discussed in Section V-E1.

E. Analysis

1) Effect of Training Set Order: We study the effect of the order of training sets as shown in Fig. 2. For Y → AGN → A and A → AGN → Y in Fig. 2(a) and (d), all the baselines exhibit a sudden decrease in accuracy on AGNews, while their performance on Yelp and Amazon is relatively good. Since the first and the third tasks are from the same domain, the chance of forgetting the first task is greatly reduced compared with the second task. As a result, all the baseline models suffer from severe forgetting on the second task, AGNews. The accuracy of OML-ER on AGNews is at least 10% lower than that on Yelp or Amazon, while other baselines have an accuracy of less than 20% on AGNews. In contrast, PMR shows a surprisingly good performance on AGNews, with its accuracy being around 33% higher than the second-best model, OML-ER. The result demonstrates PMR’s ability to ease catastrophic forgetting. For Y → A → AGN and A → Y → AGN in Fig. 2(b) and (c), all the models perform poorly, especially on the first and the second tasks. The accuracy of A-GEM, Replay and ANML-ER on Yelp and Amazon declines to almost 0. The Yelp and Amazon datasets are product reviews, while AGNews consists of news from various topics, leading to a large difference in data distribution. Experience replay only functions on the last task, AGNews. As a

\[^1\]https://pytorch.org/
result, there are insufficient epochs for the model to learn the probability distributions of previous tasks. It makes the models prone to forgetting, resulting in poor accuracy. However, even though PMR’s experience replay loss does not converge well in this case, the problem can be easily resolved by increasing the number of iterations of the last task or epochs. For AGN → Y → A and AGN → A → Y in Fig. 2(e) and (f), A-GEM, Replay, and ANML-ER perform well only in the domain of the third task, i.e., Yelp and Amazon, but suffer from catastrophic forgetting of the first task, AGNews. In contrast, OML-ER and PMR show their superiority by performing even better on AGNews than on Yelp or Amazon. This suggests that the OML framework enables fast adaptation to past examples through sparse experience replay. In addition, it can be seen in most cases that PMR’s ability to learn a task improves if the domain of that task has appeared before.

2) Effect of Replay Sample Selection: We also examine the effect of different replay sample selections, as shown in Table IV. PMR uses prototypes to represent each class and guide sample selection for memory replay, but the strategies for selecting data points through prototypes can be different. The default setting is selecting five representative samples for each class. The following three additional strategies are studied for comparison.

1) PMR\_argmax selects five examples with the maximum Euclidean distances to each prototype, i.e., five outliers.
2) PMR\_augment selects five examples from both the query set and the support set. The selection range is five times larger than the default.
3) PMR\_mix selects five representative samples and five outliers. Note that the experience replay does not include outliers from previously seen classes but only from the current class.

We evaluate all the models on the sequence Y → AGN → A. PMR\_argmax shows the worst performance due to the fact that outliers do not provide valuable discriminative information. The accuracy of PMR\_augment is also unsatisfactory, suggesting that a larger selection range may not be beneficial. Specifically, a larger selection range provides better representative samples but loses generalization. PMR\_mix achieves the highest accuracy on Yelp and Amazon. In the experience replay, the impact of representative samples from the current task is neutralized by its outliers, giving more attention to previously seen examples. PMR obtains the highest accuracy on AGNews and the highest overall accuracy. Notably, PMR shows a more than 2% accuracy improvement on Amazon after learning Yelp from the same domain. It indicates a positive knowledge transfer.
occurs within the same domain. Furthermore, all the PMR models outperform OML-ER, which uses random selection. Our sample selection strategy demonstrates its robustness even with a small memory budget (i.e., $B = 0.01\%$ of seen data). Whereas the widely used random selection scheme heavily relies on the size of the memory buffer.

3) Forgetting Evaluation: We further investigate the effect of various replay sample selection strategies on forgetting. In Table V, we define the forgetting of each task as the accuracy drop from single-task learning to CL as shown in the last column. The accuracy of single-task learning is recorded in the second column and CL accuracy is recorded in the third column. A smaller value of accuracy drop indicates less forgetting. Note that the first task and the third task are from the same domain. The accuracy of most models on the third task is lower than that on the first two tasks due to the significant difference in domain. However, it is not surprising that the accuracy of the proposed strategy tends to remember the most recent task and forget the old unigrams emphasized and others neglected. It suggests that PMR continually adjusts its selection through up-to-date prototypes. This result shows that the prototype-guided selection strategy enables sample efficiency, thereby reducing the memory budget.

4) Effect of Memory Size: We conduct this evaluation using the sequence $Y \rightarrow AGN \rightarrow A$ and the result is shown in Table VI. The accuracy improves as the number of stored samples per class increases, but the improvement stops when we increase the saved samples per class to 20. This is due to the decrease in representation power of the data points as shown in Fig. 3. The increase in $N_K$ makes the model difficult to converge.

5) Memory Insight: To verify the effectiveness of synthetic prototypes in terms of sample efficiency, we visualize the distributions of unigrams in memory for three different learning phases, as illustrated in Fig. 4. The experiment is conducted on the Yelp dataset, with a total of 25 examples in the memory, obtained after the model finishes writing. In this setting, PMR performs a total of 15 episodic experience replays. Hence, we examine the stored instances of the first, the eighth, and the last experience replay (i.e., in Episodes 1, 408, and 765), respectively. Fig. 4(a) shows a long-tailed distribution, where some unigrams appear frequently but the variety of these unigrams is small. It indicates a limited diversity of unigrams at the first learning phase. In Fig. 4(b), the unigrams’ distribution indicates the saved samples in Episode 408 cover a wider range of unigrams compared with that in Episode 1. The selected samples not only contain new unigrams but also some of the unigrams shown in Episode 1, exhibiting an improvement in sample selection. Fig. 4(c) shows a broader unigram distribution with the same number of saved examples. It also preserves some of the previously attained unigrams and introduces novel ones, with some of the old unigrams emphasized and others neglected. It suggests that PMR continually adjusts its selection through up-to-date prototypes. This result shows that the prototype-guided selection strategy enables sample efficiency, thereby reducing the memory budget.

6) Effect of Replay Rate: Fig. 5 presents the effect of the replay rate on the model’s performance. We observe a steep increase in accuracy when the replay rate $\in [0.1, 1]$. After reaching 1.0%, it shows a slight decline in accuracy as the replay rate increases. It proves that 1% experience replay is sufficient for alleviating catastrophic forgetting. However, frequent experience replay can easily lead to an overfitting problem, i.e., the model overfits to examples from memory. As a result, the model’s performance deteriorates when the replay rate is high.

7) Learning More Tasks: We evaluate models using five datasets [15] in four different orderings. For comparisons, the experimental setup follows the prior work [16], [18], in which example encoder is a pretrained BERTBASE model [43], the input sequence length is truncated to 448, and the batch size is 16. We evaluate models in three

![Fig. 3. Experience replay loss versus times of experience replay, for comparisons of different occupied memory sizes.](image-url)
Fig. 4. Visualization of unigram distribution in memory. The x-axis shows the unigram index, with a total of 1720 unigrams. The y-axis shows the number of unigrams in the range [1, 26). The data points on the x-axis indicate the number of the corresponding unigrams is 1. (a) Unigram distribution of saved samples in Episode 1. (b) Unigram distribution of saved samples in Episode 408. (c) Unigram distribution of saved samples in Episode 765. (d) Comparisons of unigram distributions in Episodes 1, 408, and 765.

Table VII

| Method     | Memory Constraint (B) | Order (1) | Order (2) | Order (3) | Order (4) | Average |
|------------|-----------------------|-----------|-----------|-----------|-----------|---------|
| A-GEM      |                       | 70.7      | 65.9      | 67.5      | 63.6      | 66.9    |
| MPA++      | 575,000(Unlimited)    | 70.8      | 70.9      | 70.2      | 70.7      | 70.6    |
| REPLAY     |                       | 69.5      | 66.2      | 65.2      | 68.3      | 67.3    |
| OML-ER     |                       | 75.4      | 76.5      | 74.5      | 75.4      | 75.7    |
| ANML-ER    |                       | 75.6      | 75.8      | 75.5      | 75.7      | 75.7    |
| OML-ER     | 165 (0.03% seen data) | 39.6      | 50.5      | 51.5      | 56.2      | 49.5    |
| PMR        |                       | 61.2      | 65.7      | 66.1      | 55.9      | 62.2    |
| OML-ER     | 1,155 (0.2% seen data)| 42.8      | 52.6      | 51.8      | 54.2      | 51.4    |
| PMR        |                       | 71.5      | 73.4      | 71.1      | 69.3      | 71.3    |

Fig. 5. Effect of replay rate on PMR using sequence Y -> AGN -> A.

different cases. Particularly, we constrain memory budgets by storing up to [5, 35, unlimited] examples per class (i.e., \( B = \{165, 1155, 575 \, 000\} \) examples). Note that PMR with unlimited memory means writing all the data to memory and replaying all the data, which is not possible for large-scale dataset and can result in out-of-memory errors. Table VII shows the accuracy of models on different sequences of five tasks. PMR shows its advantage when memory resources are extremely limited. With only 0.2% of seen data stored, PMR still achieves an impressive accuracy rate of 71.3%. The accuracy difference between PMR and the best models with unlimited memory is less than 5%. In fact, PMR even outperforms some models without the memory constraint. It demonstrates the effectiveness of PMR even under the stringent memory constraint.

3\† Results obtained from [16] and \‡ results obtained from [18].
VI. CONCLUSION

In this article, we introduce a novel CL model called PMR, which combines an OML framework with a PMR module. Specifically, we use MAML for knowledge transfer and a prototypical network as the memory replay module for knowledge retention, thereby mitigating catastrophic forgetting. Our memory replay module uses prototypical information as a reference to store previously seen samples and achieves high memory efficiency. As a result, PMR can maintain high performance with extremely limited memory budgets. The experimental results validate the superiority of PMR in terms of accuracy and efficiency. Using a relatively low replay rate and no more than 0.2% of past examples, PMR demonstrates its ability to prevent catastrophic forgetting. In addition, we show that the performance of the existing CL models heavily depends on the order of training sets. We plan to improve the stability of PMR. Furthermore, the meta-learning framework we used is the standard framework. A future research direction can be studying the effect of different meta-learning frameworks.

REFERENCES

[1] Q. Wang, J. Liu, Z. Ji, Y. Pang, and Z. Zhang, “Hierarchical correlations replay for continual learning,” Knowl.-Based Syst., vol. 250, Aug. 2022, Art. no. 109052.
[2] K. James et al., “Overcoming catastrophic forgetting in neural networks,” Proc. Nat. Acad. Sci. USA, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.
[3] Z. Li and D. Hoiem, “Learning without forgetting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[4] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 3987–3995.
[5] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, “Lifelong learning with dynamically expandable networks,” in Proc. 6th Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, Apr./May 2018.
[6] T. Adel, H. Zhao, and R. E. Turner, “Continual learning with adaptive weights (CLAW),” in Proc. 8th Int. Conf. Learn. Represent. (ICLR), Addis Ababa, Ethiopia, Apr. 2020.
[7] Q. Gao, Z. Luo, D. Klabjan, and F. Zhang, “Efficient architecture search for continual learning,” IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 2, 2022, doi: 10.1109/TNNLS.2022.3151511.
[8] H. Li, P. Barnaghi, S. Enshaeifar, and F. Ganz, “Continual learning using Bayesian neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 9, pp. 4243–4252, Sep. 2020.
[9] H. Wang, W. Xiong, M. Yu, X. Guo, S. Chang, and W. Y. Wang, “Sentence embedding alignment for lifelong relation extraction,” in Proc. Conf. North Am. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol. (NAACL-HLT), Minneapolis, MN, USA, vol. 1, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, Jun. 2019, pp. 796–806.
[10] Z. Wang, S. V. Mehta, B. Poczos, and J. Carbonell, “Efficient meta lifelong-learning with limited memory,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2020, pp. 535–548.
[11] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, “Efficient lifelong learning with A-GEM,” in Proc. 7th Int. Conf. Learn. Represent. (ICLR), May 2019.
[12] F. Sun, C. Ho, and H. Lee, “LAMOL: Language modeling for lifelong language learning,” in Proc. 8th Int. Conf. Learn. Represent. (ICLR), Addis Ababa, Ethiopia, Apr. 2020.
[13] Q. Lao, X. Jiang, M. Havaei, and Y. Bengio, “A two-stream continual learning system with variational domain-agnostic feature replay,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 9, pp. 4466–4478, Sep. 2022.
[14] C. Yan et al., “Self-weighted robust LDA for multiclassification with edge classes,” ACM Trans. Intell. Syst. Technol., vol. 12, no. 1, pp. 1–19, Feb. 2021.
[15] X. Zhang, J. J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in Proc. Adv. Neural Inf. Process. Syst., Ann. Conf. Neural Inf. Process. Syst., C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds, Montreal, QC, Canada, Dec. 2015, pp. 649–657.
[16] C. de Mason d’Autume, S. Ruder, L. Kong, and D. Yogatama, “Episodic memory in lifelong language learning,” in Proc. Adv. Neural Inf. Process. Syst., Ann. Conf. Neural Inf. Process. Syst., (NeurIPS), H. M. Wallach, H. Larochelle, A. Beygelzimer, E. D’Alché-Buc, E. B. Fox, and R. Garnett, Eds, Vancouver, BC, Canada, 2019, pp. 13122–13131.
[17] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continual learning,” in Proc. Adv. Neural Inf. Process. Syst., Ann. Conf. Neural Inf. Process. Syst., I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds, Long Beach, CA, USA, Dec. 2017, pp. 6467–6476.
[18] N. Hollia, P. Mishra, H. Yannakoudakis, and E. Shatova, “Meta-learning with sparse experience replay for lifelong language learning,” 2020, arXiv:2009.04891.
[19] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in Proc. Adv. Neural Inf. Process. Syst., Ann. Conf. Neural Inf. Process. Syst., I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds, Long Beach, CA, USA, Dec. 2017, pp. 2990–2999.
[20] R. Kemker and C. Kanan, “FearNet: Brain-inspired model for incremental learning,” in Proc. 6th Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, Apr./May 2018.
[21] A. Obamayide and A. Vlachos, “Meta-learning improves lifelong relation extraction,” in Proc. 4th Workshop Represent. Learn. NLP (RepL4NLP), 2019, pp. 224–229.
[22] X. Lu, L. Liu, L.Nie, X. Chang, and H. Zhang, “Semantic-driven interpretable deep multi-modal hashing for large-scale multimedia retrieval,” IEEE Trans. Multimedia, vol. 23, pp. 4541–4554, 2021.
[23] Y. Huang, Y. Zhang, J. Chen, X. Wang, and D. Yang, “Continual learning for text classification with information disentanglement based regularization,” in Proc. Conf. North Am. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., 2021, pp. 2736–2746.
[24] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “ICaRL: Incremental classifier and representation learning,” in Proc. IEEE Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 5533–5542.
[25] K. Lee, K. Lee, J. Shin, and H. Lee, “Overcoming catastrophic forgetting with unlabeled data in the wild,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 312–321.
[26] P. Donmez, J. G. Carbonell, and P. N. Bennett, “Dual strategy active learning,” in Proc. 18th Eur. Conf. Mach. Learn., Berlin, Germany: Springer-Verlag, 2007, pp. 116–127.
[27] T. Ramalho and M. Garnelo, “Adaptive posterior learning: Few-shot learning with a surprise-based memory module,” in Proc. 7th Int. Conf. Learn. Represent. (ICLR), New Orleans, LA, USA, May 2019.
[28] M. Toneva, A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon, “An empirical study of example forgetting during deep neural network learning,” in Proc. 7th Int. Conf. Learn. Represent. (ICLR), New Orleans, LA, USA, May 2019.
[29] T. G. J. Rudner, F. B. Smith, Q. Feng, Y. W. Teh, and Y. Gal, “Continual learning via sequential function-space variational inference,” in Proc. 39th Int. Conf. Mach. Learn., vol. 162, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., Jul. 2022, pp. 18871–18887.
[30] Y. Li, L. Zhao, K. Church, and M. Elhoseiny, “Compositional language continual learning,” in Proc. 8th Int. Conf. Learn. Represent. (ICLR), Addis Ababa, Ethiopia, Apr. 2020.
[31] Y. Li, L. Zhao, J. Wang, and J. Hestness, “Compositional generalization for primitive substitutions,” in Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), 2019, pp. 4293–4302.
[32] A. Nichol, J. Achiam, and J. Schulman, “On first-order meta-learning algorithms,” 2018, arXiv:1803.02999.
[33] M. Riemer et al., “Learning to learn without forgetting by maximizing transfer and minimizing interference,” in Proc. 7th Int. Conf. Learn. Represent. (ICLR), New Orleans, LA, USA, May 2019.
[34] K. J. Joseph and V. N. Balasubramanian, “Meta-consolidation for continual learning,” in Advances in Neural Information Processing Systems, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds, Curran Associates, 2020, pp. 14374–14386.
[35] D. A. E. Acar, R. Zhu, and V. Saligrama, “Memory efficient online meta learning,” in Proc. 35th Int. Conf. Mach. Learn., (ICML), vol. 139, M. Meila and T. Zhang, Eds., 2021, pp. 32–42.

[36] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, “Task-free continual learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 11246–11255.

[37] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in Proc. 34th Int. Conf. Mach. Learn., vol. 70, Aug. 2017, pp. 1126–1135.

[38] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2015, arXiv:1412.6980.

[39] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” in Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst., I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds. Long Beach, CA, USA, 2017, pp. 4077–4087.

[40] K. Javed and M. White, “Meta-learning representations for continual learning,” in Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst., (NeurIPS), H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, Eds. Vancouver, BC, Canada, 2019, pp. 1818–1828.

[41] S. Beaulieu et al., “Learning to continually learn,” in Proc. ECAI 24th Eur. Conf. Artif. Intell., vol. 325, G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugárin, and J. Lang, Eds., Santiago de Compostela, Spain: IOS Press, Aug. 2020, pp. 992–1001.

[42] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language representations,” in Proc. 36th Int. Conf. Learn. Represent. (ICLR), Addis Ababa, Ethiopia, Apr. 2020.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., vol. 1, Jun. 2019, pp. 4171–4186.

Stella Ho received the B.Eng. degree (Hons.) in electrical engineering from the University of New South Wales, Sydney, NSW, Australia, in 2018, and the B.Eng degree (Hons.) from Deakin University, Burwood, VIC, Australia, in 2019. She is currently pursuing the Ph.D. degree with the School of Information Technology, Deakin University, Geelong, VIC, Australia. Her current research interests include machine learning and low-resource natural language processing (NLP).

Ming Liu received the Ph.D. degree from Monash University, Clayton, VIC, Australia, in 2019. He was a Research Fellow with Deakin University, Geelong, VIC, Australia, until middle 2021, where he is currently a Lecturer with the School of Information Technology. His main research interests include natural language processing (NLP), machine learning, and large-scale data analysis.

Yong Xiang (Senior Member, IEEE) received the Ph.D. degree in electrical and electronic engineering from The University of Melbourne, Melbourne, VIC, Australia, in 2003. He is currently a Professor with the School of Information Technology, Deakin University, Geelong, VIC, Australia. His research interests include distributed computing, cybersecurity and privacy, machine learning and AI, and communications technologies. He has authored seven monographs, over 220 refereed journal articles, and over 100 conference papers in these areas.

Dr. Xiang is the Senior Area Editor of the IEEE Signal Processing Letters, the Associate Editor of the IEEE Communications Surveys and Tutorials, and the Associate Editor of Computer Standards and Interfaces. He was an Associate Editor of the IEEE Signal Processing Letters and IEEE Access, and the Guest Editor of the IEEE Transactions on Industrial Informatics and IEEE Multimedia. He has served as the honorary chair, the general chair, the program chair, the ‘TPC’ chair, the symposium chair, and the track chair for many conferences, and was invited to give keynote at numerous international conferences.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.