Learn Continually, Generalize Rapidly:
Lifelong Knowledge Accumulation for Few-shot Learning

Xisen Jin§  Bill Yuchen Lin§  Mohammad Rostami†  Xiang Ren§
§University of Southern California, †Information Sciences Institute
{xisenjin, yuchen.lin, xiangren}@usc.edu  {mrostami}@isi.edu

Abstract

The ability to continuously expand knowledge over time and utilize it to rapidly generalize to new tasks is a key feature of human linguistic intelligence. Existing models that pursue rapid generalization to new tasks (e.g., few-shot learning methods), however, are mostly trained in a single shot on fixed datasets, unable to dynamically expand their knowledge; while continual learning algorithms are not specifically designed for rapid generalization. We present a new learning setup, Continual Learning of Few-Shot Learners (CLIF), to address the challenges of both learning settings in a unified setup. CLIF assumes a model learns from a sequence of diverse NLP tasks arriving sequentially, accumulating knowledge for improved generalization to new tasks, while also retaining performance on the tasks learned earlier. We examine how the generalization ability is affected in the continual learning setup, evaluate a number of continual learning algorithms, and propose a novel regularized adapter generation approach. We find that catastrophic forgetting affects generalization ability to a lesser degree than performance on seen tasks; while continual learning algorithms can still bring considerable benefit to the generalization ability.

1 Introduction

The ability to recall acquired knowledge for learning new tasks quickly and efficiently over time has been seen as a crucial metric of general linguistic intelligence (Yogatama et al., 2019). Progress on this research problem has led to remarkable improvements in recent works on few-shot learning (Brown et al., 2020; Gao et al., 2021). However, these methods have primarily focused on learning from a static set of tasks (datasets) in an offline manner, without dynamically expanding the acquired knowledge over time. This training scheme is in contrast with the way humans process natural language (Chomsky, 2002; Montague, 1970): humans are able to process novel meanings by retaining past knowledge, combining/decomposing chunks of language into prior learned language components, and avoid learning from scratch.

Motivated by this observation, we study whether NLP models could accumulate generalizable knowledge continuously over a sequence of tasks and learn to generalize to new tasks rapidly (i.e., with few examples). This problem has not been investigated in the existing works — a related line of efforts that look to learn from sequentially arriving tasks, known as continual learning (CL) or lifelong learning (Robins, 1995; Sun et al., 2020; de Masson d’Autume et al., 2019), mainly focus on retaining the performance on seen tasks when the model is continuously updated on new tasks (i.e., to overcome the catastrophic forgetting issue).

To study this ability, we propose the Continual Learning of Few-shot Learners (CLIF) setup (illustrated in Figure 1) to simulate the challenge: In CLIF, the model learns over a sequence of NLP tasks (arriving one by one; without revisiting), and then evaluated in terms of (i) generalization to new tasks

---

1Code and data are publicly available at https://github.com/INK-USC/CLIF
(few-shot learning) tasks; and (ii) preserving its performance on solving seen tasks. We train and evaluate over a diverse set of NLP tasks, spanning over entity typing, sentiment analysis, natural language inference, and other classification tasks.

With the CLIF setup, we conduct a series of experiments on existing models, in order to understand the relationship between continuous knowledge accumulation and few-shot generalization. Our first analysis is to understand how the generalization ability evolves during continual training, and whether catastrophic forgetting affects the acquisition of generalization ability. We find a negative effect of catastrophic forgetting on the generalization ability, and a stronger negative effect on the performance over the seen tasks.

In a follow-up analysis, we find most existing CL methods hardly benefit models’ generalization ability, even they are shown to alleviate catastrophic forgetting. This implies some non-trivial challenges for accumulating knowledge that can help model generalization. Inspired by recent research on Hypernetworks for few-shot learning (Requeima et al., 2019) and continual learning approach using Hypernetworks (von Oswald et al., 2020), we propose Bi-level Hypernetworks for Adapters with Regularization to address challenges of the CLIF. We evaluate these approaches extensively by varying the number of training examples and the orders of tasks at training.

To summarize, the main contribution of this work is threefold (1) we propose CLIF setup, its data streams and protocols to comprehensively evaluate lifelong knowledge accumulation in NLP, and (2) we compare existing algorithms to demonstrate weaknesses of these algorithms (3) and propose Bi-level Hypernetworks for Adapters with Regularization as a solution to inspire future works.

2 Problem Formulation

2.1 The CLIF Problem

We assume there is an NLP model $f$ trained continually on different tasks over time (i.e., continual learning), and then rapidly generalizes to many unseen tasks with few-shot examples (i.e., few-shot adaptation). In the continual learning stage, the model encounters an ordered list of $N_u$ upstream tasks: $\{\mathcal{T}_u^1, \ldots, \mathcal{T}_u^{N_u}\}$, where each task has its own training and test sets. To test the few-shot learning ability of the sequentially trained model $f$, we then adapt it on a set of $N_c$ few-shot tasks individually

![Figure 2: Evaluations setups in CLIF. (1) and (2) measure generalization ability to new tasks; while (3) indicate forgetting on seen tasks.](image)

\[
\{\mathcal{T}_u^i\}_{i=1}^{N_u}, \text{ where only a few training examples are available for each unseen task. We name this learning setting as CLIF, which stands for continual learning for few-shot adaptation. In addition to the traditional objective in CL to preserve performance on seen tasks, in CLIF it is also crucial to retain generalizable knowledge to achieve better few-shot learning performance at the end of the training.}

**Evaluation Protocol** As illustrated in Figure 2, there are three major aspects for evaluating a method to the CLIF setting: few-shot performance, final performance, and instant performance.

1) **Few-shot Performance.** First, we evaluate the continually trained model $f$ on a set of unseen tasks, by fine-tuning it for each task $\mathcal{T}_u^i$ individually with a few annotated examples when the training over upstream tasks $\mathcal{T}_u^1 \ldots \mathcal{T}_u^{N_u}$ ends. Thus, we can assess the few-shot generalization ability. We note the few-shot accuracy for a task $\mathcal{T}_u^i$ as $s_{FS}^i = F(\hat{Y}_v^i, \hat{Y}_u^i)$, where $\hat{Y}_v^i$ is the predictions over the test examples of task $\mathcal{T}_u^i$, $\hat{Y}_u^i$ is the set of ground truth labels, and $F$ is the metric function (e.g., accuracy). We report $s_{FS}$ averaged over all few-shot tasks, i.e., $s_{FS} = \frac{1}{N_c} \sum_{i=1}^{N_c} s_{FS}^i$. We also compute a relative improvement $\Delta_{FS} = \frac{s_{FS} - s_{FS}'}{s_{FS}}$ over the performance $s_{FS}'$ of the models separately trained on each few-shot task.

2) **Instant Performance.** We evaluate the performance of an upstream task $\mathcal{T}_u^i$ right after the model $f$ finishes the learning on it. We note the set of model prediction on the test set of task $\mathcal{T}_u^i$ right after the model $f$ learns the task $j$ as $\hat{Y}_u^{i,j}$. The instant performance over task $\mathcal{T}_u^i$ is defined as $s_{inst}^i = F(\hat{Y}_u^{i,j}, \hat{Y}_u^i)$. For example, we evaluate the performance of $f$ on $\mathcal{T}_u^2$ after the model $f$ is trained on the data of $\mathcal{T}_u^1$ and $\mathcal{T}_u^2$ before further train it on $\mathcal{T}_u^3$. The performance of $f$ on $\mathcal{T}_u^2$ now can thus tell us how well the model transfers its knowledge from learning $\mathcal{T}_u^1$ to learn $\mathcal{T}_u^2$ — using
We additionally compute a relative improvement
the full list of tasks in Appendix A.

To push the CLIF challenge to a more practical
the CLIF setting is particularly chal-
Challenges

The CLIF setting is particularly chal-
Computations

3) **Final Performance.** We also evaluate the performance of \( f \) at the end of the continual learning over upstream tasks to know how much the model \( f \) forgets the knowledge about the task after it learns to solve more tasks. The final accuracy \( s_{\text{final}}^{i} \) of a task \( T_{u}^{i} \) is defined as \( F(Y_{i}^{u}, \hat{Y}_{i}^{u}; N_{u}) \). Similarly, we report the averaged final accuracy over all tasks, noted as \( s_{\text{final}} = \frac{1}{N_{u}} \sum_{i=1}^{N_{u}} s_{\text{final}}^{i} \). For a single model, the forgetting can be quantified as \( s_{\text{inst}} - s_{\text{final}} \).

**Challenges** The CLIF setting is particularly chal-
most few-shot learning methods assume that the upstream training datasets for all tasks are always available and there is no temporal order for learning. Hence, the upstream tasks can be learned jointly in a multi-task learning setting. However, the CLIF problem follows a continual learning setup, where the tasks are visited sequentially without revisiting. Thus, methods relying on random sampling from a task distribution are not applicable.

### 2.2 Tasks and Data Streams

To push the CLIF challenge to a more practical
we use the GLUE (Wang et al., 2020) bench-

<table>
<thead>
<tr>
<th>Learning Stage</th>
<th>Tasks</th>
<th># Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continual ( T_{c} )</td>
<td>GLUE (Wang et al., 2019a)</td>
<td>( N_{u} = 9 )</td>
</tr>
<tr>
<td>Few-shot ( T_{f} )</td>
<td>DivFSL (Bansal et al., 2020)</td>
<td>( N_{v} = 17 )</td>
</tr>
</tbody>
</table>

| CLIF-26                  | SuperGLUE-RTE, TweetEval-MRPC, Sentiment, Scicite, GLUE-MRPC, Scitail, KILT-Fever, ... |
| CLIF-55                  | SuperGLUE-CB, Dbpedia-14, NER, ... |

Table 1: **Overview of datasets employed for up-

To adopt it for our learning setting, we specify
We also consider alternative task orders in our ex-
We use 90 examples per class for contin-
As the test labels for GLUE are not publicly available, we report perfor-
We convert regression tasks (e.g. STS-B) to binary classification tasks by
For CLIF-55, we use 90 examples per class for contin-
We use \( k = 16 \) examples per class in few-shot learning tasks for both CLIF-26 and CLIF-55 if not specified, and include more setups of \( k \) in the experiments. As the test labels for GLUE are not publicly available, we report performance on validation sets. We convert regression tasks (e.g. STS-B) to binary classification tasks by setting the threshold in the middle of the maximum and minimum regression scores.

All examples are converted into sequence-
to-sequence question-answering formats following (McCann et al., 2018) to allow a single model to solve all tasks. We consider exact match between the generated answer span and the ground-truth span as a correct prediction. For both the upstream tasks and few-shot tasks in CLIF-26 and CLIF-55, we use the prediction accuracy as the metric function.

### 3 Method

This section presents baseline methods to set up the lower bounds for the CLIF problem, and approaches to improve the performance. We view an approach by its base model and the learning algorithm. We first introduce the base models in our study (Sec. 3.1); Then, we introduce a few existing methods for continual learning and continual meta-

---

2https://huggingface.co/datasets
regularized bi-level adapter generation framework to better address the CLIF problem (Sec. 3.3).

3.1 Base NLP Models

BART and BART-Adapter. As we formulate the NLP tasks in the CLIF problem in a unified text-to-text format, we use pre-trained language models (LMs) as the architecture of the model \( f \) and fine-tune the entire model during training. We mainly use the BART-base (Lewis et al., 2020) model for our experiments. We also include Adapter training (Houlsby et al., 2019) as an alternative to fine-tuning the entire BART model. Here, adapters (Houlsby et al., 2019) are two-layer Multi-Layer Perceptrons (MLPs) plugged after each layer of BART. Given the output \( h_l \) at the \( \ell \)-th layer of the transformer, the adapted output is computed as \( h'_l = h_l + f^p(h_l) \), where \( f^p \) is the adapter layer at layer \( \ell \). Only adapters are learned during training, while the BART model is frozen. We note two approaches BART and BART-Adapter respectively.

Hyper-Networks for Adapter Generation. In addition to BART and BART-Adapter, we also use consider a HyperNetwork (HNet) architecture. The hypernetwork, noted as \( g \), takes a task representation \( z \) as input and generates model parameter of another prediction model, noted as \( f \) to solve the task. In few-shot learning, \( z \) is usually computed as the average representation of training examples of the task, \( z = \frac{1}{|\mathcal{D}_t^i|} \sum_{(x_j, y_j) \in \mathcal{D}_t^i} f_c(x_j, y_j) \), where \( \mathcal{D}_t^i \) is the training set of the task \( T^i \) and \( f_c \) in an encoder model. In our case, we use a BART model as \( f_c \) and feed it the concatenation of \( x \) and label \( y \) in text format to obtain the task representation \( z \). As the model allows flexible control of model parameters with training examples, it is broadly applied for few-shot learning (Requeima et al., 2019; Gidaris and Komodakis, 2018); besides, \( z \) can also be randomly initialized and end-to-end learned (Ha et al., 2017). As the parameter space of large-scale PTLMs like BART is huge, following (Ye and Ren, 2021), we generate model parameters only for adapters.

In summary, we consider BART fine-tuning, BART-Adapter learning and HNet for adapter generalization as three base NLP models. In section 3.2, we introduce algorithms to learn these models in the CLIF setting.

3.2 Baseline Learning Algorithms

Single Task Learning To understand the reference performance of a base model on an upstream task without any knowledge transfer, we apply the single task learning (STL) method, which trains and tests a model \( f \) on the dataset of each task in isolation. In this case, we ignore the sequential nature of the CLIF problem so we can use this STL performance to assess the effectiveness of different continual methods (introduced below). Ideally, a valid CL algorithm should have a better few-shot accuracy than STL results, meaning that it accumulates knowledge and effectively transfer it for learning. Similarly, to know the reference performance of the few-shot tasks, we learn a model \( f \) for each few-shot task on the given examples, without any upstream training, so that we can use such performance to assess how well a CLIF method improves the generalization ability.

Continual Learning Algorithms As a straightforward baseline method, we use Vanilla to denote simply training the model \( f \) sequentially on the upstream tasks. Specifically, it trains the model \( f \) on \( T^i_u \) until its performance converges and then continually train \( f \) on the data of \( T^{i+1}_u \). Note that the access of the data on previous tasks is not allowed in CL. We also consider CL algorithms such as EWC (Kirkpatrick et al., 2017), MbPA++ (de Masson d’Autume et al., 2019) and meta-MbPA (Wang et al., 2020) in our experiments. We use an online variant of EWC (Schwarz et al., 2018). EWC regularizes the change of important model parameters during training. The MbPA++ method performs test-time adaptation over a few training examples stored in the memory. The Meta-MbPA method includes a meta-learning objective to adapt fast.

As a comparator that does not suffer from forgetting, we also report the results of multi-task learning over upstream tasks (MTL) for reference.

Hyper-Networks for CL. von Oswald et al. (2020) proposed a hypernetwork-based continual learning algorithm, where the high-level idea of mitigating catastrophic forgetting is to penalize the hypernetwork for the change of generated model weights for previous tasks when it learns a new task. While the original work generates entire parameters of a model, we adapt it to PTLMs by generating the weights of adapters only. We note the approach as HNet-Reg.
Specifically, when the model has just finished learning the task $T_{u-1}^i$ and right before learning the task $T_u^i$ in the continual learning stage, we compute the adapter weights generated by our current hypernetwork for all prior tasks $T_{u-1}^i, T_{u-1}^{i-1}, \ldots, T_{u-1}^1$ — where the generation is controlled by applying the hypernetwork $h$ on the stored task representations of previous tasks $1 \ldots i - 1$, noted as $M = \{z_{h_1}, \ldots, z_{h_{i-1}}\}$. Here, the task representation $z_i$ for task $T_u^i$ is randomly initialized before learning the task and optimized jointly while learning the task. Then, in each step of learning $T_u^i$, we randomly sample a prior task $T_u^j (j < i)$ to regularize the hypernetwork learning. It penalizes the $L_2$ distance between the adapter weights generated at the current step $\theta_j$ and the pre-computed one, i.e., $||\theta_j - \theta_j^{i-1}||^2_2$. Therefore, we avoid the hypernetwork $g$ changes its output for a prior task too much during the continual learning stage, so that the knowledge accumulation is better guaranteed for the learned model.

**Limitations.** EWC and HNET-Reg are not well-designed for the CLIF problem, which additionally tries to improve the few-shot generalization on unseen tasks after continual learning. While the test-time adaptation in MbPA and meta-MbPA may benefit few-shot learning, such ability is not studied in these works. Besides, as these two algorithms store real examples of previous training tasks, it is not applicable in privacy sensitive applications where data from earlier task is no longer accessible, which is a typical scenario in continual learning.

### 3.3 Our Extension: Bi-level Hypernetworks for Adapters with Regularization

Inspired by hypernetwork approaches for few-shot learning and continual learning, we extend the hypernetwork-based CL methods for CLIF. We present a novel method, Bi-level Hypernetwork for Adapters with Regularization (BiHNet+Reg), which learns to use the bi-level task representations to generate adapter weights for learning a fast adaptive model over a sequence of tasks, while mitigating the forgetting effect via regularization.

As shown in Figure 3, the proposed method consists of three components: (1) a context predictor to generate bi-level task representations (i.e., high-resource and few-shot representations) from training examples, (2) a hypernetwork to generate weights of adapters given the task representations, and (3) a regularization term to discourage weight changes of seen tasks to avoid forgetting following (von Oswald et al., 2020). We discuss each individual component below.

**Context Predictor.** We propose to generate two task representations for each task $t$ to model it in the high-resource and few-shot cases respectively, denoted as $z_h^t$ and $z_f^t$, with a frozen BART model. The high-resource representations are used to encourage the knowledge transfer during continual learning; the few-shot task representations help us mimic the few-shot tasks in the few-shot learning stage for better generalization, similar to meta-learning. Specifically, we use an LM (e.g., BART) as the context representation model $R$ for encoding an example $(x, y)$: we feed $x$ and $y$ to the encoder and the decoder of the model $R$, and use the latent representation from this last-layer activation. The high-resource task representation is then computed as the average of all examples’ representations in task $t$, noted as $z_h^t = \frac{1}{|D_t|} \sum_{(x, y) \in D_t} R(x, y)$; while the few-shot task representation $z_f^t$ uses the average of a limited number (say, $K$) of sampled examples $z_f^t = \frac{1}{K} \sum_{(x, y) \in \Gamma(D_t, K)} R(x, y)$, where $\Gamma(D_t, K)$ means sampled $K$ examples in $D_t$.

Note that the high-resource representations of upstream tasks are stored in a memory module over time during the continual learning, $M = \{z_h \mid t \in \{T_u\}_{i=1}^{N_u}\}$. In the few-shot learning stage, we set $K$ as the number of given examples, so the $z_h = \ldots$
$z_f$ for any tasks.

**Adapter-Wise Hypernetworks.** Following the practice introduced in Sec. 3.1, we use a hypernetwork $g$ to generate weights of adapters between the layers of the frozen BART model $f$. During training, we use high-resource and sampled task representations $z^h_t$ and $z^f_t$ to generate adapter weights separately, noted as $\theta^h_t$ and $\theta^f_t$. We optimize the prediction loss with both adapters.

**Regularization.** Given that the HyperNetwork is the only trainable part in our model, we impose regularization on generated adapters to mitigate forgetting following HNet+Reg introduced in 3.2. While our BiHNet is trained to generate adapters from both high-resource and low-resource task representations, we find it sufficient to only store and regularize outputs from high-resource task representations.

**Summary and Highlights**

To sum up, our proposed method first generates bi-level task representations for training adapter-wise hypernetworks with a regularization term dedicated for avoiding forgetting over time. Unlike replay-memory based CL approaches (e.g., MbPA (de Masson d’Autume et al., 2019)), our method does not store any real training examples. Instead, it uses task representations for storing the memory, and thus allows the method to be applied in privacy-sensitive scenarios.

4 Results and Analysis

We address our two major research questions in this section: (1) how models accumulate generalizable knowledge over time in a CL setup compared to offline setups given potential catastrophic forgetting, and (2) whether continual learning approaches reduce catastrophic forgetting of both seen-task performance and generalizable knowledge. We experiment with various combinations of model architectures in 3.1 and learning algorithms 3.2. We note a method by its model architecture and CL algorithm applied, e.g., BART-Vanilla, BiHNet-EWC. We include details of implementation in Appendix A.

4.1 Examining Knowledge Accumulation

In this section, we present analysis of model’s ability to acquire generalizable knowledge in offline and CL setup. We note BiHNet methods, which correspond to learning to generate adapters, should be compared with BiHNet-Single and BART-Adapter-Single, which are zero-knowledge baselines that learns to generate or learn adapters from random initialization; similarly, BART methods should be compared with BART-Single. We focus on identifying challenges in CLIF, and leave discussions of methodology in the next subsection.

**Q1: Is knowledge from upstream tasks helpful for a model’s few-shot generalization in offline and continual learning setups?** To answer the question, we compare the performance of MTL with learning separate models per few-shot task without learning upstream tasks. Table 2 summarizes the results. On both CLIF-26 and CLIF-55 datasets, we see BiHNet-MTL could outperform zero-knowledge baselines in few-shot Acc. by 0.4% and 1.0%, which implies upstream tasks are helpful for few-shot generalization in standard offline learning setups. For BART models, we notice BART-MTL improves over BART-Single on CLIF-55 datasets by 2.5%. However, we notice the opposite for CLIF-26. Given that the entire BART parameters are optimized in these models, we hypothesize that BART-MTL may have suffered from the forgetting of knowledge in the pre-trained BART model itself; while in adapter and BiHNet models, the BART model is frozen. Therefore, in the rest of the section, we focus more on BiHNet approaches.

**Q2: How does the model’s generalization ability evolve over time?** We focus on BiHNet-Vanilla and BART-Vanilla approaches and answer three sub-questions.

Is the knowledge being monotonically accumulated over upstream tasks? In comparison to two zero-knowledge baselines, we notice BiHNet-Vanilla generally improves both Instant Accuracy (4.2% on CLIF-26 and 6.8% on CLIF-55) and Few-shot Accuracy (0.8% on CLIF-55), except...
Few-shot Acc. - 2.5% -2.4% -7.8% - 69.32 -2.0% -10.5% - 1.1% 55.22 - 47.04 -3.1% -10.9% - -9.7% 56.16 4.2% -0.4% 44.03 2.7% -1.1% - -

Table 2: Final accuracy (Final Acc.) and instant accuracy (Inst. Acc.) over upstream tasks and accuracy over few-shot learning tasks (Few-shot Acc.) on CLIF-26 and CLIF-55 tasks. We compute relative improvement of instant accuracy (ΔInst.) and few-shot accuracy (ΔFS.) over zero-knowledge baselines (the better one between BART-Adapter-Single and BiHNet-Single for BiHNet, and BART-Single for BART approaches).3

in few-shot Acc. on CLIF-26 (-0.4%). The results confirm positive knowledge accumulation to some extent. In Figure 4, we plot the few-shot accuracy on CLIF-26 when the model sequentially visits each upstream training task. We note the few-shot accuracy of BiHNet-Vanilla does not monotonically increase, which implies interference between these upstream learning tasks or forgetting of generalizable knowledge.

*Does the order of the tasks matter?* Figure 5 presents performance of methods under different orders of tasks on CLIF-26. We order the tasks by increasing and decreasing relevance to few-shot learning tasks, where the relevance is defined as a few shot accuracy when the model transfers from a single upstream task. The results show in both orders BiHNet-Vanilla is less competitive than BART-Adapter-Single. It implies that in continual learning the knowledge accumulation is less robust without CL algorithms.

**Q3: Does model’s catastrophic forgetting hinder its knowledge accumulation?** In Table 2, we see clear differences between final accuracy of Vanilla and MTL approaches (by around 20 points), which verifies the catastrophic forgetting of seen-task performance when training examples are not i.i.d. However, we find the gap between MTL and Vanilla training is close for few-shot learning performance, where BART-Vanilla is even better than BART-MLT, which can be a positive outcome of adequate forgetting for alleviating overfitting (Wang et al., 2020). It indicates the catastrophic forgetting influence generalization ability to a lesser degree compared to its effect on seen-task performance.

Figure 5: Few-shot learning performance of BiHNet-Vanilla and BiHNet-Reg on CLIF-26 tasks when training tasks are presented in different orders.

4.2 Effect of Continual Learning Algorithms

With the insights obtained for earlier questions, we now analyze whether baseline continual learning algorithms and the proposed approach help knowledge accumulation and improve models’ (few-shot) generalization ability.

**Q1: Do continual learning algorithms mitigate catastrophic forgetting?** From Table 2, we notice MbPA++, meta-MbPA, EWC clearly improve final accuracy over BART-Vanilla or BiHNet-Vanilla on CLIF-26, which confirm positive effects on mitigating catastrophic forgetting. On CLIF-55, which features much more training tasks and less examples per tasks, we find baseline CL algorithms fail to improve final accuracy. For memory-based approaches such as MbPA++ and meta-MbPA, it can because of significant overfit-
few-shot accuracy of trainable embeddings is slightly higher on CLIF-26 and CLIF-55. We observe BiHNet-Reg always achieves the best performance and the improvement is generally more significant when the training sets are smaller.

Discussion. Our results indicate BiHNet-Reg could effectively improve knowledge accumulation over time compared to similar adapter learning frameworks (BiHNet-Single and BART-Adapter-Single). However, BiHNet-Reg does not rival BART-Single in terms or few-shot learning accuracy. We believe this is due to the restricted model capacity of adapter, as compared to fine-tuning entire transformer. This opens up future work on improving continual learning algorithms that are compatible with PTLM fine-tuning.

5 Related Work

Continual Learning. The primary challenge that is addressed in CL literature is overcoming catastrophic forgetting. Generally, existing CL methods encompass memory and generative replay-based approaches (Robins, 1995; Lopez-Paz and Ranzato, 2017; Shin et al., 2017), regularization-based approaches (Kirkpatrick et al., 2017; Nguyen et al., 2018) and model expansion based approaches (Shin et al., 2017). Recently, continual learning has drawn attention in the NLP field (Sun et al., 2020; Wang et al., 2019b; Huang et al., 2021).

Continual Meta-Learning. There exists literature that studies continual meta-learning outside NLP application, with various definition of the...
problem. Some prior works (Xu et al., 2019; de Masson d’Autume et al., 2019; Wang et al., 2020) aim to develop algorithms that allows fast recovery of previous performance when a few training examples of an early task are available again at the test time. Caccia et al. (2020) proposed a setup where models visit a sequence of potentially re-occurring tasks and measured online cumulative performance as metrics. Antoniou et al. (2020) assumes the model visits a sequence of few-shot classification tasks while the test tasks consist of seen classes at training. The problem setup of Jerfel et al. (2019) is most related to ours which learns to perform few-shot learning on new tasks better, but is only studied for image classification tasks with much smaller number tasks. To our best knowledge, our work is the first to study continual knowledge accumulation for few-shot learning in diverse NLP tasks for large-scale transformer models.

6 Conclusion

We present the Continual Learning of Few-Shot Learners (CLIF) challenge to simulate the scenario where a learner continually accumulate (generalizable) knowledge over a sequence of NLP tasks, while retaining its performance on the seen tasks. We propose evaluation protocols to study the performance of existing continual learning algorithm, and present our method BiHNet-Reg. We demonstrate the potentials of building a NLP system that, through continual training, can perform more tasks and also become more efficient in mastering new tasks. Future works include extending our work to task agnostic scenarios where the distribution of data may shift continuously and studying algorithms for continual refinement of large-scale pre-trained models with emerging unlabeled data.

Acknowledgements

This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600007, the DARPA MCS program under Contract No. N660011924033, the Defense Advanced Research Projects Agency with award W911NF-19-20271, NSF IIS 2048211, NSF SMA 1829268, and gift awards from Google, Amazon, JP Morgan and Sony. We would like to thank all the collaborators in USC INK research lab for their constructive feedback on the work.

References


Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019b. Sen-


Table 4: Statistics of trainable and total model parameters in each model to learn 9 GLUE tasks.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trainable Params</th>
<th>Total Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>BART</td>
<td>139M</td>
<td>139M</td>
</tr>
<tr>
<td>BART-Adapter</td>
<td>72M</td>
<td>212M</td>
</tr>
<tr>
<td>BiHNet</td>
<td>260M</td>
<td>405M</td>
</tr>
<tr>
<td>BiHNet$_d=4$</td>
<td>40M</td>
<td>180M</td>
</tr>
</tbody>
</table>

Table 5: Performance when we use a smaller hidden dimension ($d=4$) for the HyperNet in BiHNet-Reg.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BART-Adapter-Single</td>
<td>-</td>
<td>74.98 ±0.17</td>
<td>59.00 ±1.9</td>
</tr>
<tr>
<td>BiHNet$_d=4$, Reg</td>
<td>72.50 ±2.3</td>
<td>79.45 ±0.5</td>
<td>60.68 ±1.0</td>
</tr>
</tbody>
</table>

A Implementation Details

We tune hyperparameters except the time steps of few-shot training on the validation set of upstream continual learning tasks. We tune the hyperparameters on CLIF-26 and apply the same for CLIF-55 for the same approaches. We tune learning rates by enumerating over [3e-4, 1e-4, 3e-5, 1e-5], and finally use a learning rate of 3e-5 for all MTL approaches and fine-tuned BART approaches (e.g., BART-EWC, BART-Vanilla), and a learning rate of 1e-4 for BiHNet, HNet, and BART-Adapter-Single. We use a batch size of 64 across experiments. We train the model for at most 100 epochs for each training task with a patience of 3 epochs without validation performance improvement. Before training on a new task, we revert the model to the checkpoint with the best validation performance in the previous task. In the few-shot learning stage, we use the same learning rate and train the model for 400 epochs, assuming no validation sets to perform early stopping. The number of training steps are decided based on the performance of BiHNet-Vanilla on airline, conll, and disaster tasks. We set the hidden size of adapters inserted between layers of BART transformers as 256 and the one in the classification head as 64. The weight generator in BiHNet is implemented as a two-layer MLP with a hidden size of 32. For replay based approaches (MbPA++ and meta-MbPA), we store all examples following these works and randomly draw mini-batches to replay every 100 training steps. For BiHNet, HNet, and EWC, we set the regularization strength (coefficient before the regularization loss term) as 0.01 without further tuning. We use a sample size 64 to compute the few-shot task representation on CLIF-26 and 10 for CLIF-55 at training. Experiments are run on Nvidia Quadro 6000 or Quadro 8000 GPUs with cuda version 10.1 installed. Through out the experiments (including the hyperparameter search), we run each method with three random seeds.

Details of Datasets. For CLIF-26, we use the train, validation, and test split from Bansal et al. (2020). For a seen trained model, we evaluate its few-shot ability over 5 different partitions of train-test splits of a single few-shot task. For CLIF-55, we use the train, validation, and test sets provided in the datasets library4. The few-shot training and validation sets are random samples of the official train and validation splits; while we do not subsample the test split. Similarly, we evaluate few-shot learning ability over 5 different samples of training and validation examples.

Details of Task Orders. Table 7 summarize the list of 45 upstream training tasks and 10 few-shot training tasks. Table 6 further shows the order of continual learning tasks.

B Parameter Efficiency

We show the statistics of trainable and total parameters in each compared architecture in Table 4 on CLIF-26. In our default settings, BiHNet has twice as many trainable parameters as BART and above three times as BART-Adapter. However, we could significantly reduce the number of parameters by setting the hidden size $d$ of the Hypernetwork smaller than the number of the tasks. We reduce $d$ to 4, and summarize the results in 5. We notice the approach achieves instant accuracy and few-shot accuracy on par with BiHNet-Reg in the standard setup. We notice the approach achieves lower final accuracy compared to the default setup, but the score is still more competitive than baselines, such as BART-MbPA and BART-meta-MbPA, and BiHNet-Vanilla.

4https://huggingface.co/datasets
<table>
<thead>
<tr>
<th>Task Order</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>cola, sst2, mrpc, qqp, stsb, mnli, qnli, wnl, rte</td>
</tr>
<tr>
<td>Relevance ↓</td>
<td>mnli, sst2, qqp, qnli, stsb, mrpc, cola, rte, wnl</td>
</tr>
<tr>
<td>Relevance ↑</td>
<td>wnl, rte, cola, mrpc, stsb, qnli, qqp, stt2, mnli</td>
</tr>
</tbody>
</table>

| Default    | ai2_arc, aqua_rat, boolq, codah, commonsense_qa, cosmos_qa, dream, eli5-askh, eli5-asks, eli5-eli5, freebase_qa, hel-laswag, jeopardy, kilt_hotpotqa, kilt_nq, kilt_trex, kilt_zsre, lama-conceptnet, lama-google_re, lama-squad, lama-trex, math_qa, me_taco, numer_sense, openbookqa, qasc, quail, quarel, quartz-no_knowledge, quartz-with_knowledge, race-high, race-middle, sciq, search_qa, social_i_qa, squad-no_context, superglue-copa, superglue-multirc, swag, web_questions, wino_grande, wiqa |

Table 6: Order of continual learning tasks in CLIF-26 and CLIF-55 datasets.
<table>
<thead>
<tr>
<th>Task Name</th>
<th>Task</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ade_corpus_v2-classification</td>
<td>other</td>
<td>Gurulingappu et al. 2012</td>
</tr>
<tr>
<td>circa</td>
<td>other</td>
<td>Louis et al. 2020</td>
</tr>
<tr>
<td>discovery</td>
<td>other</td>
<td>Sileo et al. 2019</td>
</tr>
<tr>
<td>emotion</td>
<td>emotion</td>
<td>Saravia et al. 2018</td>
</tr>
<tr>
<td>ethos-directed_vs_generalized</td>
<td>hate speech detection</td>
<td>Mollas et al. 2020</td>
</tr>
<tr>
<td>ethos-disability</td>
<td>hate speech detection</td>
<td>Mollas et al. 2020</td>
</tr>
<tr>
<td>ethos-gender</td>
<td>hate speech detection</td>
<td>Mollas et al. 2020</td>
</tr>
<tr>
<td>ethos-sexual_orientation</td>
<td>other</td>
<td>Warstadt et al. 2019</td>
</tr>
<tr>
<td>glue-cola</td>
<td>nli</td>
<td>Williams et al. 2018</td>
</tr>
<tr>
<td>glue-mnli</td>
<td>paraphrase</td>
<td>Dolan and Brockett 2005</td>
</tr>
<tr>
<td>glue-qnli</td>
<td>nli</td>
<td>Rajpurkar et al. 2016</td>
</tr>
<tr>
<td>glue-sst2</td>
<td>sentiment analysis</td>
<td>Socher et al. 2013</td>
</tr>
<tr>
<td>glue-wnli</td>
<td>nli</td>
<td>Levesque et al. 2012</td>
</tr>
<tr>
<td>google_wellformed_query</td>
<td>other</td>
<td>Faruqui and Das 2018</td>
</tr>
<tr>
<td>hate_speech_offensive</td>
<td>hate speech detection</td>
<td>Davidson et al. 2017</td>
</tr>
<tr>
<td>hateplain</td>
<td>fact checking</td>
<td>Mathew et al. 2020</td>
</tr>
<tr>
<td>health_fact</td>
<td>sentiment analysis</td>
<td>Kotsyna and Toni 2020</td>
</tr>
<tr>
<td>imdb</td>
<td>fact checking</td>
<td>Thorne et al. 2018</td>
</tr>
<tr>
<td>liar</td>
<td>fact checking</td>
<td>Wang 2017</td>
</tr>
<tr>
<td>onestop_english</td>
<td>other</td>
<td>Vajjala and Lučić 2018</td>
</tr>
<tr>
<td>paws</td>
<td>paraphrase</td>
<td>Zhang et al. 2019</td>
</tr>
<tr>
<td>rotten_tomatoes</td>
<td>sentiment analysis</td>
<td>Pang and Lee 2005</td>
</tr>
<tr>
<td>scicite</td>
<td>other</td>
<td>Cohan et al. 2019</td>
</tr>
<tr>
<td>scitail</td>
<td>nli</td>
<td>Khot et al. 2018</td>
</tr>
<tr>
<td>sick</td>
<td>nli</td>
<td>Marelli et al. 2014</td>
</tr>
<tr>
<td>sms_spam</td>
<td>other</td>
<td>Almeida et al. 2011</td>
</tr>
<tr>
<td>superglue-wic</td>
<td>other</td>
<td>Pilehvar and Camacho-Collados 2019</td>
</tr>
<tr>
<td>superglue-wsc</td>
<td>other</td>
<td>Levesque et al. 2012</td>
</tr>
<tr>
<td>trec</td>
<td>other</td>
<td>Li and Roth 2002; Hovy et al. 2001</td>
</tr>
<tr>
<td>trec_finegrained</td>
<td>other</td>
<td>Li and Roth 2002; Hovy et al. 2001</td>
</tr>
<tr>
<td>tweet_eval-emoji</td>
<td>emotion</td>
<td>Barbieri et al. 2020</td>
</tr>
<tr>
<td>tweet_eval-emotion</td>
<td>emotion</td>
<td>Barbieri et al. 2020</td>
</tr>
<tr>
<td>tweet_eval-irony</td>
<td>emotion</td>
<td>Barbieri et al. 2020</td>
</tr>
<tr>
<td>tweet_eval-offensive</td>
<td>emotion</td>
<td>Barbieri et al. 2020</td>
</tr>
<tr>
<td>tweet_eval-sentiment</td>
<td>emotion</td>
<td>Barbieri et al. 2020</td>
</tr>
<tr>
<td>tweet_eval-stance_abortion</td>
<td>emotion</td>
<td>Barbieri et al. 2020</td>
</tr>
<tr>
<td>tweet_eval-stance_climate</td>
<td>emotion</td>
<td>Barbieri et al. 2020</td>
</tr>
<tr>
<td>tweet_eval-stance_hillary</td>
<td>emotion</td>
<td>Barbieri et al. 2020</td>
</tr>
<tr>
<td>wiki_auto</td>
<td>other</td>
<td>Jiang et al. 2020</td>
</tr>
<tr>
<td>yahoo_answers_topics</td>
<td>topic</td>
<td>(link)</td>
</tr>
<tr>
<td>superglue-ch</td>
<td>nli</td>
<td>de Marneffe et al. 2019</td>
</tr>
<tr>
<td>dbpedia_14</td>
<td>topic</td>
<td>Lehmann et al. 2015</td>
</tr>
<tr>
<td>wiki_qa</td>
<td>other</td>
<td>Yang et al. 2015</td>
</tr>
<tr>
<td>emo</td>
<td>emotion</td>
<td>Chatterjee et al. 2019</td>
</tr>
<tr>
<td>yelp_polarity</td>
<td>sentiment analysis</td>
<td>Zhang et al. 2015; (link)</td>
</tr>
<tr>
<td>ethos-religion</td>
<td>hate speech detection</td>
<td>Mollas et al. 2020</td>
</tr>
<tr>
<td>financial_phrasebank</td>
<td>sentiment analysis</td>
<td>Malo et al. 2014</td>
</tr>
<tr>
<td>tab_fact</td>
<td>fact checking</td>
<td>Chen et al. 2020</td>
</tr>
<tr>
<td>anli</td>
<td>nli</td>
<td>Nie et al. 2020</td>
</tr>
<tr>
<td>ethos-race</td>
<td>hate speech detection</td>
<td>Mollas et al. 2020</td>
</tr>
</tbody>
</table>

Table 7: Datasets and tasks included in **CLIF-55** for upstream training and few-shot learning.