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, 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 less degree than performance on seen tasks; while continual learning algorithms can still bring considerable benefit to the generalization ability[1].

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., 2020). 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.

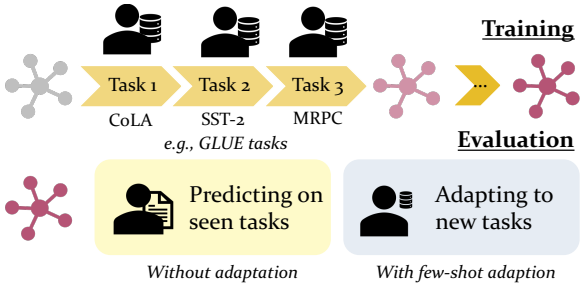

Figure 1: **Overview of the Training and Evaluation setup in CLIF.** The model learns over a number of training tasks sequentially and is evaluated over all the seen tasks. We also evaluate its ability to adapt to new tasks with only a small number of labeled examples.

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 (few-shot learning) tasks; and (ii) preserving its performance on solving seen tasks. We train and

[1]Code and data are publicly available at https://github.com/INK-USC/CLIF

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 (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 *u*pstream 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_v *few-shot* tasks individually $\{\mathcal{T}_v^i\}_{i=1}^{N_v}$, 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 training.

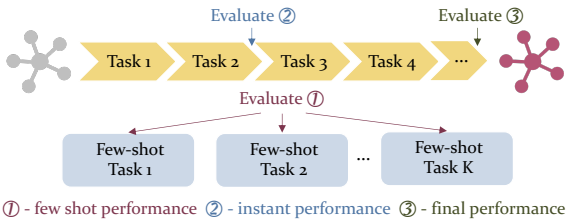

Figure 2: **Evaluations setups in CLIF**. (1) and (2) measure generalization ability to new tasks; while (3) indicate forgetting on seen tasks.

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 $\mathcal{T}_v^i$ individually with a few annotated examples when the training over upstream tasks $\mathcal{T}_u^1..\mathcal{T}_u^{N_u}$ ends. Thus, we can assess the *few-shot generalization* ability. We note the few-shot accuracy for a task $\mathcal{T}_v^i$ as $s_{\text{FS}}^i = F(\mathcal{Y}_v^i, \hat{\mathcal{Y}}_v^i)$, where $\hat{\mathcal{Y}}_v^i$ is the predictions over the test examples of task $\mathcal{T}_v^i$, $\mathcal{Y}_v^i$ 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.*, $s_{\text{FS}} = \frac{1}{N_v}\sum_{i=1}^{N_v} s_{\text{FS}}^i$. We also compute a relative improvement Δ_{FS} over 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 $\hat{\mathcal{Y}}_u^{i,j}$. The instant performance over task $\mathcal{T}_u^i$ is defined as $s_{\text{inst.}}^i = F(\mathcal{Y}_u^i, \hat{\mathcal{Y}}_u^{i,i})$. 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 the performance when f is trained only on $\mathcal{T}_u^2$ as a reference. We compute average instant performance of all upstream tasks, $s_{\text{inst.}} = \frac{1}{N_u}\sum_{i=1}^{N_u} s_{\text{inst.}}^i$. We additionally compute a relative improvement

Learning Stage	Tasks	# Tasks
	CLIF-26	
Continual ($\mathcal{T}_u$)	GLUE (Wang et al., 2019a)	N_u =9
Few-shot ($\mathcal{T}_v$)	DivFSL (Bansal et al., 2020)	N_v =17
	CLIF-55	
Continual ($\mathcal{T}_u$)	SuperGLUE-RTE, TweetEval-Sentiment, Scicite, GLUE-MRPC, Scitail, KILT-Fever, ...	N_u =45
Few-shot ($\mathcal{T}_v$)	SuperGLUE-CB, Dbpedia-14, Wiki-QA, emo, Yelp-Polarity, ethos-religion, tab-fact, financial-phrasebank, ANLI, ethos-race	N_v =10

Table 1: **Overview of datasets employed for upstream continual training and few-shot learning**. We include the full list of tasks in Appendix A.

$\Delta_{\text{Inst.}}$ over models separately trained on each upstream task to indicate benefit of upstream learning.

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^i_{final} of a task $\mathcal{T}^i_u$ is defined as $F(\mathcal{Y}^i_u, \hat{\mathcal{Y}}^{i,N_u}_u)$. Similarly, we report the averaged final accuracy over all tasks, noted as $s_{\text{final}} = \frac{1}{N_u}\sum_{i=1}^{N_u} s^i_{\text{final}}$. For a single model, the forgetting can be quantified as $s_{\text{inst}} - s_{\text{final}}$.

Challenges The CLIF setting is particularly challenging for existing few-shot learning methods. 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 an *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 setup, we consider a diverse set of NLP tasks to perform CL and few shot learning. We consider two dataset combinations, referred to as `CLIF-26` and `CLIF-55` tasks, summarized in Table 1. In the first combination, following Bansal et al. (2020), we use the GLUE (Wang et al., 2019a) benchmark as our upstream tasks for CL stage for experiments which consists of $N_u = 9$ tasks. We then evaluate the few-shot learning ability over $N_v = 17$ DivFSL (Bansal et al., 2020) tasks, spanning over diverse NLP tasks including sentiment analysis, entity typing, natural language inference. In `CLIF-55`, we train and test the model over $N_u = 45$ and $N_v = 10$ tasks selected from Huggingface datasets library[2]. The selected datasets span over a broad family of NLP tasks, including natural language inference, emotion classification, topic classification, fact checking, hate speech detection, paraphrasing, and others.

To adopt it for our learning setting, we specify an order on the tasks presented to the model for `CLIF-26` and `CLIF-55` (details in Appendix A). We also consider alternative task orders in our experiments. The model sequentially visits each task during training. We limit the number of training examples in each GLUE tasks in `CLIF-26` to 10,000 to avoid overly imbalanced datasets. For `CLIF-55`, we use 90 examples per class for continual learning. 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.

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-learning (Sec. 3.2). Finally, we present a novel 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 train-

[2] https://huggingface.co/datasets

ing. 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 ℓ-th layer of the transformer, the adapted output is computed as $h'_\ell = h_\ell + f^a_\ell(h_\ell)$, where f^a_ℓ is the adapter layer at layer ℓ. 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 generate model parameter of another prediction model, noted as f to solve the task. In few-shot learning, z is usually computed as a the average representation of training examples of the task, $\boldsymbol{z} = \frac{1}{|\mathcal{D}^i_{tr}|}\sum_{(\boldsymbol{x}_j,\boldsymbol{y}_j)\in\mathcal{D}^i_{tr}} f_e(\boldsymbol{x}_j, \boldsymbol{y}_j)$, where $\mathcal{D}^i_{tr}$ is the training set of the task $\mathcal{T}^i$ and f_e in an encoder model. In our case, we use a BART encoder as f_e and feed it the concatenation of $\boldsymbol{x}$ and label $\boldsymbol{y}$ 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, in literature studying outside few-shot learning, $\boldsymbol{z}$ can be randomly initialized and end-to-end learned (Ha et al., 2016). 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 $\mathcal{T}^i_u$ until its performance converges and then continually train f on the data of $\mathcal{T}^{i+1}_u$. 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) are considered in our experiments. 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 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. 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 $\mathcal{T}^{i-1}_u$ and right before learning the task $\mathcal{T}^i_u$ in the continual learning stage, we store the adapter weights generated by our *current* hypernetwork for all prior tasks $\mathcal{T}^1_u..\mathcal{T}^{i=1}_u$, noted as $\{\hat{\boldsymbol{\theta}}^{i-1}_1, \hat{\boldsymbol{\theta}}^{i-1}_2, \ldots, \hat{\boldsymbol{\theta}}^{i-1}_{i-1}\}$ — where the generation is controlled by applying the hypernetwork h on the stored task representations of previous tasks $1..i-1$, noted as $\mathcal{M} = \{\boldsymbol{z}^1_h, \ldots, \boldsymbol{z}^{i-1}_h\}$. Here, the task representation $\boldsymbol{z_i}$ for task $\mathcal{T}^i_u$ is randomly initialized before learning the task and optimized jointly while

learning the task. Then, in each step of learning $\mathcal{T}_u^i$, we randomly sample a prior task $\mathcal{T}_u^j$ $(j < i)$ to regularize the hypernetwork learning. It penalizes the ℓ_2 distance between the adapter weights generated at the current step $\boldsymbol{\theta}_j$ and the pre-computed one, *i.e.*, $||\boldsymbol{\theta}_j - \hat{\boldsymbol{\theta}}_j^{i-1}||_2^2$. 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 (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 $\boldsymbol{z}_h^t$ and $\boldsymbol{z}_f^t$, with a frozne BART encoder. The high-resource representations are used for encourage the knowledge transfer during continual learning; the few-shot task representations help us mimic the few-shot tasks in

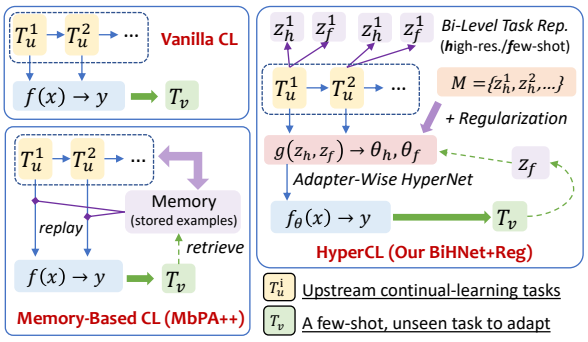

Figure 3: **A comparison between different typical continual methods to the CLIF problem**. The Vanilla CL method simply trains the model on a sequence of tasks $\mathcal{T}_u$. Memory-based methods such as MbPA++ (de Masson d'Autume et al., 2019) store a small set of examples of prior tasks and then replay them during learning. Our BiHNet+Reg method uses a hypernetwork to generate the weights of model adapters according to bi-level (high-resource and few-shot) task representations.

the few-shot learning stage for better generalization, similar to meta-learning. The **high-resource** task representation is then computed as the average of **all** examples' context vectors[3] in task t, noted as $\boldsymbol{z}_h^t = \frac{1}{|\mathcal{D}_t|}\sum_{(\boldsymbol{x}_i,\boldsymbol{y}_i)\in\mathcal{D}_t} R(\boldsymbol{x}_i,\boldsymbol{y}_i)$; while the few-shot task representation $\boldsymbol{z}_f^t$ uses the average of a limited number (say, K) of **sampled** examples $\boldsymbol{z}_f^t = \frac{1}{K}\sum_{(\boldsymbol{x}_i,\boldsymbol{y}_i)\in\Gamma(\mathcal{D}_t,K)} R(\boldsymbol{x}_i,\boldsymbol{y}_i)$, where $\Gamma(\mathcal{D}_t, K)$ means sample K examples in $\mathcal{D}_t$.

Note that the high-resource representations of upstream tasks are stored in a memory module over time during the continual learning, $\mathcal{M} = \{\boldsymbol{z}_h^t | t \in \{\mathcal{T}_u^i\}_{i=1}^{N_u}\}$. In the few-shot learning stage, we set K as the number of given examples, so the $\boldsymbol{z}_h = \boldsymbol{z}_f$ 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 $\boldsymbol{\theta}_t^h$ and $\boldsymbol{\theta}_t^f$. We optimize the prediction loss with both adapters.

Regularization. Given that the HyperNetwork is the only trainable parts in our model, we impose regularization on generated adapters to mitigate forgetting following HNet+Reg introduced in 3.2.

[3]Specifically, we use an LM (e.g., BART) as the context representation model R for encoding an example $(\boldsymbol{x}, \boldsymbol{y})$, by concatenating $\boldsymbol{x}$ and $\boldsymbol{y}$ as a single input sentence to R and using the latent representation from it last-layer activation.

CLIF Dataset	**CLIF 26** (GLUE → DivFSL)					**CLIF 55** (Classification)				
Methods ↓ Metrics →	Final Acc.	Inst. Acc.	F-S Acc.	$\Delta_{\text{Inst.}}$	$\Delta_{\text{FS.}}$	Final Acc.	Inst. Acc.	F-S Acc.	$\Delta_{\text{Inst.}}$	$\Delta_{\text{FS.}}$
Single Task-Learning										
BART-Single	-	$79.39_{\pm 0.7}$	$60.99_{\pm 0.5}$	-	-	-	$69.32_{\pm 0.3}$	$68.49_{\pm 0.7}$	-	-
BART-Adapter-Single	-	$74.98_{\pm 0.7}$	$59.00_{\pm 1.9}$	-	-	-	$65.15_{\pm 0.5}$	$65.70_{\pm 0.8}$	-	-
BiHNet-Single	-	$76.67_{\pm 0.4}$	$52.66_{\pm 0.9}$	-	-	-	$66.44_{\pm 0.2}$	$64.57_{\pm 1.1}$	-	-
Continual learning										
BART-Vanilla	$19.73_{\pm 0.2}$	$79.92_{\pm 0.2}$	$58.96_{\pm 3.2}$	0.6%	-3.3%	$49.46_{\pm 1.7}$	$71.26_{\pm 0.6}$	$66.08_{\pm 0.6}$	2.8%	-3.5%
BART-MbPA++	$59.52_{\pm 1.0}$	$77.48_{\pm 0.5}$	$56.26_{\pm 1.4}$	-2.4%	-7.7%	$51.75_{\pm 1.5}$	$67.18_{\pm 1.0}$	$61.03_{\pm 3.5}$	-3.1%	-10.9%
BART-meta-MbPA	$55.69_{\pm 0.9}$	$78.63_{\pm 0.5}$	$57.88_{\pm 1.0}$	-0.9%	-5.1%	$51.55_{\pm 2.3}$	$67.92_{\pm 1.2}$	$61.30_{\pm 2.0}$	-2.0%	-10.5%
BiHNet-Vanilla	$53.15_{\pm 2.1}$	$79.90_{\pm 0.2}$	$58.76_{\pm 1.6}$	6.5%	-0.4%	$44.03_{\pm 1.7}$	$70.97_{\pm 1.6}$	$66.23_{\pm 0.6}$	8.9%	0.8%
BiHNet-EWC	$56.15_{\pm 1.6}$	$78.73_{\pm 0.3}$	$58.36_{\pm 1.7}$	5.0%	-1.1%	$7.15_{\pm 2.1}$	$72.43_{\pm 1.0}$	$58.08_{\pm 0.8}$	11.1%	-11.6%
BiHNet-Reg	$77.22_{\pm 1.1}$	$80.24_{\pm 0.4}$	$60.09_{\pm 1.1}$	7.0%	1.9%	$56.16_{\pm 1.6}$	$73.04_{\pm 0.6}$	$68.46_{\pm 0.2}$	12.1%	4.2%
Multi-Task Learning										
BART-MTL	$74.07_{\pm 0.4}$	-	$55.02_{\pm 2.5}$	-	-9.7%	$63.78_{\pm 0.0}$	-	$70.20_{\pm 0.4}$	-	2.5%
BiHNet-MTL	$78.20_{\pm 0.3}$	-	$59.22_{\pm 0.8}$	-	0.4%	$64.93_{\pm 0.0}$	-	$66.40_{\pm 3.6}$	-	1.0%
Majority	55.22	-	47.04	-	-	52.74	-	59.52	-	-

Table 2: Final accuracy (Final Acc.) and instant accuracy (Instant 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 ($\Delta_{\text{Inst.}}$) and few-shot accuracy (Δ_{FS}) over zero-knowledge baselines (the better one between BART-Adapter-Single and BiHNet-Single for BiHNet, and BART-Single for BART approaches).[4]

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 hypothize hat 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

[4]Note that, for single-task learning baselines, "Inst. Acc." column is used to refer to the averaged accuracy of individual models trained for each upstream task.

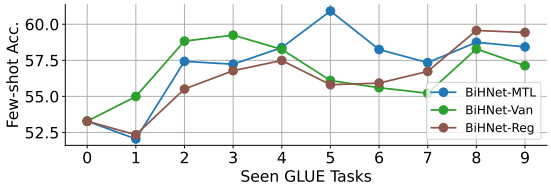

Figure 4: Few-shot learning performance on CLIF-26 test tasks evaluated after each checkpoint of the model as the model sequentially visit upstream continual learning tasks.

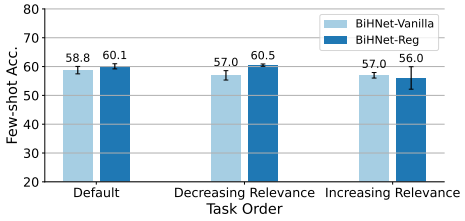

Figure 5: Few-shot learning performance of BiHNet-Vanilla and BiHNet-Reg on CLIF-26 tasks when training tasks are presented in different orders.

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.

Does the knowledge being monotonically accumulated over upstream tasks? In comparison to two zero-knowledge baselines, we notice BiHNet-Vanilla generally improves both Instant Accuracy (6.5% on CLIF-26 and 8.9% on CLIF-55) and Few-shot Accuracy (0.8% on CLIF-55), except in few-shot Acc. on CLIF-26 (-0.4%). The results confirms 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 present 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 few shot accuracy when the model transfers from a single upstream tasks. 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-MTL, which can be a positive outcome of adequate forgetting for alleviating overfitting (Wang et al., 2020). It indicates the catastrophic catastrophic forgetting influence generalization ability to a less degree compared to its effect on seen-task performance.

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 overfitting to stored examples. In contrast, BiHNet-Reg is effective in both datasets.

Q2: Does mitigating catastrophic forgetting better retain generalization ability? By comparing the few-shot accuracy of BiHNet-Vanilla and BiHNet-Reg, we notice an improvement of few-shot accuracy and instant accuracy by 1.9% and 4.2% on two datasets. From Figure 5, we see BiHNet-Reg outperforms BiHNet-Vanilla in the default and decreasing relevance order; while we observe an outlier in BiHNet-Reg runs in the increasing relevance order. From Figure 4, we see few-shot learning accuracy improves more stable as BiHNet-Reg learns more upstream tasks.

Q3: Does BiHNet-Reg improve over HNet-Reg? The major differences of BiHNet-Reg compared to HNet-Reg (Oswald et al., 2020) are (1) few-shot task task representations and (2) inferring task representations with context predictors instead of learning them as trainable embeddings. As an

	CLIF 26		CLIF 55	
	Final Acc.	Few-shot Acc.	Final Acc.	Few-shot Acc.
BiHNet-Reg	$77.22_{\pm 1.1}$	$60.09_{\pm 1.1}$	$56.16_{\pm 1.6}$	$68.46_{\pm 0.2}$
-Few-shot TR	$78.78_{\pm 1.3}$	$59.01_{\pm 0.6}$	$55.90_{\pm 1.4}$	$68.13_{\pm 0.5}$
+Train Embs	$65.50_{\pm 1.5}$	$61.60_{\pm 0.1}$	$44.87_{\pm 0.1}$	$66.14_{\pm 0.2}$

Table 3: Ablation study on BiHNet-Reg: after removing few-shot task-representations (-Short-term TR), and replacing context predictors with trainable embeddings (+Train Embs.).

ablation study, we progressively replace out two components in BiHNet , as shown in Table 3. We see removing few-shot task-representation causes the few-shot accuracy to drop on both datasets by 1.08 and 0.33 points; while replacing the context predictor with trainable task embedding caused a clear drop of final accuracy by more than 10 points. We notice the few-shot accuracy of trainable embeddings is slightly higher on CLIF-26 by 1.5 points, but lower on CLIF-55 by 2.3 points which has more upstream training tasks.

Q4: Sensitivity Analysis: how do models perform under various number of few-shot training examples. Figure 6 summarizes few-shot performance of different methods under different number of training examples per class on CLIF-26. We observe BiHNet-Reg always achieves the best performance and the improvement is 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 for 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 tackle 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).

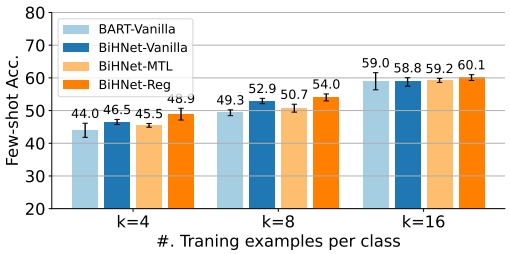

Figure 6: Few-shot learning performance of BART-Vanilla, BiHNet-Vanilla, BiHNet-MTL, and BiHNet-Reg under different number of training examples per class ($k = 4, 8, 16$) on CLIF-26. See Figure 7 in Appendix for CLIF-55 results.

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) aims to develop algorithms that allows fast recovery of previous performance when a few training examples of an early task is available again at the test time. Caccia et al. (2020) proposed a setup where models visit a sequence of potentially re-occuring 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 data.

References

Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*, DocEng '11, page 259–262, New York, NY, USA. Association for Computing Machinery.

Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. 2020. Defining benchmarks for continual few-shot learning. *arXiv preprint arXiv:2004.11967*.

Trapit Bansal, Rishikesh Jha, and A. McCallum. 2020. Learning to few-shot learn across diverse natural language classification tasks. In *COLING*.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*, volume 6, pages 6–4. Venice.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

T. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. Henighan, R. Child, A. Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Page-Caccia, Issam Hadj Laradji, Irina Rish, Alexandre Lacoste, David Vázquez, et al. 2020. Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning. *Advances in Neural Information Processing Systems*, 33.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. SemEval-2019 task 3: EmoContext contextual emotion detection in text. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 39–48, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Noam Chomsky. 2002. *Syntactic structures*. Walter de Gruyter.

Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. 2019. Structural scaffolds for citation intent classification in scientific publications. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3586–3596, Minneapolis, Minnesota. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. In *NeurIPS*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Manaal Faruqui and Dipanjan Das. 2018. Identifying well-formed natural language questions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 798–803, Brussels, Belgium. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *ArXiv*, abs/2012.15723.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Spyros Gidaris and Nikos Komodakis. 2018. Dynamic few-shot visual learning without forgetting. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4367–4375.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*, 45(5):885–892. Text Mining and Natural Language Processing in Pharmacogenomics.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

N. Houlsby, A. Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and S. Gelly. 2019. Parameter-efficient transfer learning for nlp. In *ICML*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization. *arXiv preprint arXiv:2104.05489*.

Ghassen Jerfel, E. Grant, T. Griffiths, and K. Heller. 2019. Reconciling meta-learning and continual learning with online mixtures of tasks. In *NeurIPS*.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural CRF model for sentence alignment in text simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7943–7960, Online. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. In *AAAI*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, and Others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Neema Kotonya and Francesca Toni. 2020. Explainable automated fact-checking for public health claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7740–7754, Online. Association for Computational Linguistics.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, D. Kontokostas, Pablo N. Mendes, Sebastian Hellmann, M. Morsey, Patrick van Kleef, S. Auer, and C. Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, page 552–561. AAAI Press.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pages 6467–6476.

Annie Louis, Dan Roth, and Filip Radlinski. 2020. "I'd rather just go to bed": Understanding indirect answers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7411–7425, Online. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *J. Assoc. Inf. Sci. Technol.*, 65(4):782–796.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 216–223, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. Hatexplain: A benchmark dataset for explainable hate speech detection. *arXiv preprint arXiv:2012.10289*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. 2020. Ethos: an online hate speech detection dataset. *ArXiv*, abs/2006.08328.

Richard Montague. 1970. Universal grammar. *1974*, pages 222–46.

Cuong V Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. 2018. Variational continual learning. *ICLR*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

J. Oswald, C. Henning, J. Sacramento, and Benjamin F. Grewe. 2020. Continual learning with hypernetworks. *ICLR*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

James Requeima, J. Gordon, John Bronskill, Sebastian Nowozin, and R. Turner. 2019. Fast and flexible multi-task classification using conditional neural adaptive processes. In *NeurIPS*.

Anthony Robins. 1995. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *NeurIPS*, pages 2990–2999.

Damien Sileo, Tim Van De Cruys, Camille Pradel, and Philippe Muller. 2019. Mining discourse markers for unsupervised sentence representation learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3477–3486, Minneapolis, Minnesota. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Fan-Keng Sun, Cheng-Hao Ho, and Hung yi Lee. 2020. Lamol: Language modeling for lifelong language learning. In *ICLR*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Sowmya Vajjala and Ivana Lučić. 2018. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 297–304, New Orleans, Louisiana. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019b. Sentence embedding alignment for lifelong relation extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 796–806.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2:*

Short Papers), pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Zirui Wang, S. Mehta, B. Póczos, and J. Carbonell. 2020. Efficient meta lifelong-learning with limited memory. *EMNLP*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Hu Xu, Bing Liu, Lei Shu, and Philip Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Qinyuan Ye and X. Ren. 2021. Zero-shot learning by generating task-specific adapters. *ArXiv*, abs/2101.00420.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA. MIT Press.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

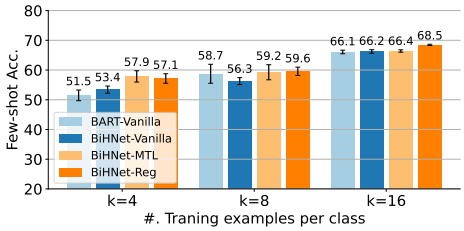

Figure 7: Few-shot learning performance of BART-Vanilla, BiHNet-Vanilla, BiHNet-MTL, and BiHNet-Reg under different number of training examples per class ($k = 4, 8, 16$) on CLIF-55.

	Trainable Params	Total Params
BART	139M	139M
BART-Adapter	72M	212M
BiHNET	266M	405M
BiHNET$_{d=4}$	40M	180M

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

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 hyperpameters 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-tuend 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 600 steps ($k \in \{16, 8\}$) or 400 steps ($k = 4$), 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.

Methods	Final Acc.	Inst. Acc.	F-S Acc.
BART-Adapter-Single	-	74.98 ± 0.7	59.00 ± 1.9
BiHNet$_{d=4}$-Reg	72.50 ± 2.3	79.45 ± 0.5	60.68 ± 1.5

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

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 splits provided in the datasets library[5]. 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,

[5]https://huggingface.co/datasets

Task Order	Tasks
	CLIF-26
Default	cola, sst2, mrpc, qqp, stsb, mnli, qnli, wnli, rte
Relevance ↓	mnli, sst2, qqp, qnli, stsb, mrpc, cola, rte, wnli
Relevance ↑	wnli, rte, cola, mrpc, stsb, qnli, qqp, sst2, mnli
	CLIF-55
Default	ai2_arc, aqua_rat, boolq, codah, commonsense_qa, cosmos_qa, dream, eli5-askh, eli5-asks, eli5-eli5, freebase_qa, hellaswag, jeopardy, kilt_hotpotqa, kilt_nq, kilt_trex, kilt_zsre, lama-conceptnet, lama-google_re, lama-squad, lama-trex, math_qa, mc_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.

but the score is still more competitive than baselines, such as BART-MbPA and BART-meta-MbPA, and BiHNet-Vanilla.

Task Name	Task	Reference
Upstream tasks		
ade_corpus_v2-classification	other	Gurulingappa et al. 2012
circa	other	Louis et al. 2020
discovery	other	Sileo et al. 2019
emotion	emotion	Saravia et al. 2018
ethos-directed_vs_generalized	hate speech detection	Mollas et al. 2020
ethos-disability	hate speech detection	Mollas et al. 2020
ethos-gender	hate speech detection	Mollas et al. 2020
ethos-sexual_orientation	hate speech detection	Mollas et al. 2020
glue-cola	other	Warstadt et al. 2019
glue-mnli	nli	Williams et al. 2018
glue-mrpc	paraphrase	Dolan and Brockett 2005
glue-qnli	nli	Rajpurkar et al. 2016
glue-qqp	paraphrase	(link)
glue-rte	nli	Dagan et al. 2005; Bar-Haim et al. 2006 Giampiccolo et al. 2007; Bentivogli et al. 2009
glue-sst2	sentiment analysis	Socher et al. 2013
glue-wnli	nli	Levesque et al. 2012
google_wellformed_query	other	Faruqui and Das 2018
hate_speech_offensive	hate speech detection	Davidson et al. 2017
hatexplain	hate speech detection	Mathew et al. 2020
health_fact	fact checking	Kotonya and Toni 2020
imdb	sentiment analysis	Maas et al. 2011
kilt_fever	fact checking	Thorne et al. 2018
liar	fact checking	Wang 2017
onestop_english	other	Vajjala and Lučić 2018
paws	paraphrase	Zhang et al. 2019
rotten_tomatoes	sentiment analysis	Pang and Lee 2005
scicite	other	Cohan et al. 2019
scitail	nli	Khot et al. 2018
sick	nli	Marelli et al. 2014
sms_spam	other	Almeida et al. 2011
superglue-rte	nli	Dagan et al. 2005; Bar-Haim et al. 2006 Giampiccolo et al. 2007; Bentivogli et al. 2009
superglue-wic	other	Pilehvar and Camacho-Collados 2019
superglue-wsc	other	Levesque et al. 2012
trec	other	Li and Roth 2002; Hovy et al. 2001
trec-finegrained	other	Li and Roth 2002; Hovy et al. 2001
tweet_eval-emoji	emotion	Barbieri et al. 2020
tweet_eval-emotion	emotion	Barbieri et al. 2020
tweet_eval-irony	emotion	Barbieri et al. 2020
tweet_eval-offensive	emotion	Barbieri et al. 2020
tweet_eval-sentiment	emotion	Barbieri et al. 2020
tweet_eval-stance_abortion	emotion	Barbieri et al. 2020
tweet_eval-stance_climate	emotion	Barbieri et al. 2020
tweet_eval-stance_hillary	emotion	Barbieri et al. 2020
wiki_auto	other	Jiang et al. 2020
yahoo_answers_topics	topic	(link)
Few-shot learning tasks		
superglue-cb	nli	de Marneffe et al. 2019
dbpedia_14	topic	Lehmann et al. 2015
wiki_qa	other	Yang et al. 2015
emo	emotion	Chatterjee et al. 2019
yelp_polarity	sentiment analysis	Zhang et al. 2015; (link)
ethos-religion	hate speech detection	Mollas et al. 2020
financial_phrasebank	sentiment analysis	Malo et al. 2014
tab_fact	fact checking	Chen et al. 2020
anli	nli	Nie et al. 2020
ethos-race	hate speech detection	Mollas et al. 2020

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