Lifelong Learning of Few-shot Learners across NLP Tasks

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

Abstract

Recent advances in large pre-trained language models have greatly improved the performance on a broad set of NLP tasks. However, adapting an existing model to new tasks often requires (repeated) re-training over enormous labeled data that is prohibitively expensive to obtain. Moreover, models learned on new task may gradually "forget" about the knowledge learned from earlier tasks (*i.e.*, catastrophic forgetting). In this paper, we study the challenge of lifelong learning to few-shot learn over a sequence of diverse NLP tasks, through continuously fine-tuning a language model. We investigate the model's ability of few-shot generalization to new tasks while retaining its performance on the previously learned tasks. We explore existing continual learning methods in solving this problem and propose a continual meta-learning approach which learns to generate adapter weights from a few examples while regularizing changes of the weights to mitigate catastrophic forgetting. We demonstrate our approach preserves model performance over training tasks and leads to positive knowledge transfer when the future tasks are learned[1].

1 Introduction

Recent advances in NLP stems from emergence of language models that are pretrained on huge and large-scale corpus (Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019) and then are fine-tuned to train a model for a given downstream task. Despite dramatic success of this approach for single task learning (STL), *e.g.*, sentiment analysis (Xu et al., 2019) and text classification (Sun et al., 2019a), fine-tuning is inefficient when the goal is to learn multiple diverse downstream tasks that arrive in a stream (Biesialska et al., 2020). the model may suffer from catastrophic forgetting (Robins, 1995), *i.e.*, significant performance drops on previous tasks, which would necessitate model retraining. While with STL, models may perform poorly due to limited training data. 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 combining/decomposing chunks of language into prior learned language components, avoiding learning from scratch, and retains past knowledge. This means that knowledge about language is accumulated over time by without any interference and may benefit future learning.

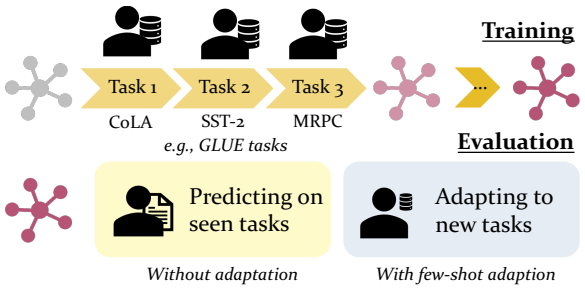

Figure 1: **Overview of Training and Evaluation setup in CLIF.** The model learns over a number of training tasks sequentially. It is then evaluated over all seen tasks, and its ability to adapt to new tasks with only a small number of labeled examples.

Inspired by the ability of humans, a desirable alternative solution for NLP is to enable a single base model to expand its knowledge continually to learn the new future tasks, ideally accelerated through exploiting the obtained knowledge from past experiences. This learning scenario is referred as the continual learning (CL) or lifelong machine learning (LML) setting (Parisi et al., 2019). Works on addressing CL for NLP domain tasks are limited (Sun et al., 2020; Wang et al., 2020; Huang et al., 2021); besides, existing works only focus on reducing catastrophic forgetting of training tasks, while overlook data-efficient adaptation to new tasks. We propose the the Continual LearnIng of

[1]Code: https://github.com/INK-USC/CLIF

Few-shot Learners (CLIF) setup (illustrated in Figure 1) to deal with both challenges: In CLIF, the model learns over a sequence of high-resource natural language tasks, and is evaluated for both solving seen tasks and fast adaptation to new few-shot learning tasks. We follow Bansal et al. (2020) to train and evaluate over a diverse set of tasks: the training datasets consist of 9 GLUE tasks, and the few-shot evaluation consists of 17 tasks spanning over entity typing, sentiment analysis, and other classification tasks. Mitigating catastrophic forgetting is still a challenge in CLIF; while learning to few-shot adapt is a unique challenge for algorithms that relies on random sampling from a distribution of tasks (Finn et al., 2017a), given that only one training task can be accessed at the same time. It makes the challenge beyond combining an existing continual learning algorithm with a few-shot learning algorithm.

We address the following research questions in our experiments: (1) can continually trained models retain performance effectively on seen tasks? (2) can models transfer knowledge to learn new tasks better, and (3) can the resulting continually trained models learn new tasks with only a few training examples? In addition to evaluating existing methods, we also present a novel Long-short term Hypernet (HNET) architecture to dynamically generate adapters (Houlsby et al., 2019) to be plugged in between the layers of transformers, associated with regularization over the generated adapters to alleviate forgetting. We demonstrate that not only catastrophic forgetting is addressed but performance across the sequential training and few shot tasks improve as the result of positive knowledge transfer from the past learned tasks.

To summarize, our contributions are that (1) we propose CLIF setup, its data streams and evaluation protocols to better study language learning for sequential tasks over time, and (2) we compare existing algorithms to demonstrate weaknesses of these algorithms (3) and propose HNET with regualarization as a solution to inspire future works.

2 Lifelong Learning of Few-shot Learner

Our primary goal is to obtain a system that continually learns over NLP tasks without catastrophic forgetting, and accumulate knowledge to solve future tasks (potentially over limited training examples). We present a formal problem definition and evaluation protocols for CLIF (Sec. 2.1), and the datasets used for our study (Sec. 2.2).

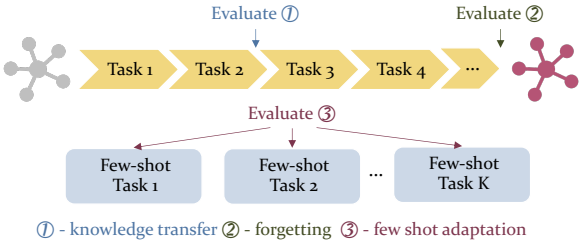

Figure 2: **Evaluations setups in CLIF**. Each evaluation focus on one specific ability of the model.

2.1 Problem Setup

We consider that a prediction model f (built upon a pre-trained language model) encounters a sequence of NLP training tasks $\{\mathcal{T}_{tr}^i\}_{i=1}^T$ (noted as sequential training tasks). Each task is associated with a training dataset $\mathcal{D}^i = \{(\boldsymbol{x}_j^i, \boldsymbol{y}_j^i)\}_{j=1}^{N_{tr}^i}$, where $\boldsymbol{x}$ is the model input (*e.g.*, a sentence) and $\boldsymbol{y}$ is the prediction target. The model f sequentially visits each dataset. The performance is evaluated over test sets $\mathcal{D}_{te}^i = \{\boldsymbol{x}_j^i\}_{j=1}^{N_{te}^i}$ for each task i. In addition, we assume a separate set of few shot learning tasks, noted as $\{\mathcal{T}_f^k\}_{k=1}^K$, associated with few-shot training and testing sets $\mathcal{D}^k$ and $\mathcal{D}_{te}^k$.

The goal is equip the model with three primary abilities: (i) the ability of continually learning sequential training tasks while mitigating catastrophic forgetting; (ii) the ability to transfer knowledge while learning new tasks (iii) the ability to perform few-shot adaptation on new tasks. Thus, these three abilities are evaluated in our experiments. These evaluation setups are illustrated in Figure 2, and are discussed more in details below.

Continual Learning: We evaluate performance on a training task $\mathcal{T}_{tr}^i$ (1) right after learning the task $\mathcal{T}_{tr}^i$ and (2) after learning the all the training tasks $\{\mathcal{T}_{tr}^i\}_{i=1}^T$ with f. The former evaluates the model's ability to accumulate knowledge from old tasks to learn new tasks. The latter evaluates the effect of catastrophic forgetting on performance on the past learned tasks. We assume the task identifier is provided at both training and the test time.

Few-Shot Adaptation on New Tasks: After learning all (or a portion of) sequential training tasks $\{\mathcal{T}_{tr}^i\}_{i=1}^T$, we adapt the model f separately to each few-shot learning task $\mathcal{T}_f^k$ with their training sets and evaluate on test sets.

Since the tasks arrive sequentially, learning the tasks jointly, i.e., multi-task learning, is not feasi-

	Number of Tasks
Sequential Training Tasks	
GLUE	9
Few-shot Learning Tasks	
Entity Typing	2
Text Classification	10
Natural Language Inference	1
Sentiment Analysis	4

Table 1: Overview of datasets employed for sequential training and few-shot learning.

ble. Moreover, few-shot learning methods that rely on random sampling from a task distribution is not feasible either, as only one task is available in a given time. Although continual learning algorithm is still necessary, existing algorithms do not consider few-shot learning ability. While a few study on continual meta-learning methods exist (Wang et al., 2020; He et al., 2019), they focus on recalling performance of seen tasks with test-time adaptation instead of seeking to generalize to new tasks. Therefore, our problem setup creates a new challenge for existing algorithms.

2.2 Tasks and Data Streams

We use the GLUE (Wang et al., 2019) in our experiments. The GLUE dataset consists of nine natural language understanding tasks, including, single-sentence tasks CoLA and SST-2, similarity and paraphrasing tasks, MRPC, STS-B and QQP, and natural language inference tasks MNLI, QNLI, RTE and WNLI. The dataset is broadly used to study knowledge transfer across related NLP tasks. To adopt it within our learning setting, we consider an order on the tasks encountered by the model, following how they are listed on the GLUE official website[2]. We will study the effect of the task ordering in future works. The model sequentially visits each tasks during training. We limit the number of training examples in each GLUE task to 10,000 to avoid overly imbalanced datasets. As the test labels for GLUE is not publicly available, we report performance on validation sets. All examples are converted into question-answering formats following (McCann et al., 2018) to allow a single model to solve all tasks, and only exact match of the generated answer sequence and the ground truth counts as a correct prediction.

To evaluate few-shot learning ability, we employ the 17 few-shot learning tasks introduced by Bansal et al. (2020)[3], spanning over entity typing, text classification, natural language inference, and sentiment analysis. Each task has only 16 labeled examples per class.

2.3 Evaluation Metrics

For both datasets, we use prediction accuracy as the metrics. We report accuracy scores on each sequential training task evaluated (1) right after learning each task and (2) after learning all tasks. The two scores are referred to as Instant Accuracy and Final Accuracy respectively. We also report Few-shot Accuracy scores on each few-shot training task after few-shot adaptation,

3 Method

We identify existing approaches can be applied to address the CLIF challenge. We apply continual learning algorithms to learn sequential training tasks, then fine-tune them over each few-shot learning task. Alternatively, we apply existing continual-meta learning algorithms. The performance is compared against several lower-bounds of performance. We also propose our own approach, Long-Short Term Hypernetwork with Regularization, to simutaneously address the challenge of continual and few-shot learning. We introduce the compared baselines in Sec. 3.1, and our own approach in Sec. 3.2.

3.1 Baselines

We consider methods from three categories, namely Single-Task Learning, Continual Learning, and Continual Meta-Learning.

Single Task Learning: We report performance of the model on individual tasks when the tasks are learned in isolation. Improved performance over this baseline demonstrates the effect of *knowledge transfer* from past tasks when learning new tasks. For single-task learning, we report the number of training an adapter (Houlsby et al., 2019) for BART (Lewis et al., 2020) for each individual task. In case of positive knowledge transfer, STL serves as a lower-bound of performance evaluated right after learning each task in continual learning.

Continual Learning Algorithms: We report performance of the model when the tasks are learned sequentially without using any mechanism to tackle catastrophic forgetting (Vanilla online training). This will serve as lower-bound to

[2] https://gluebenchmark.com/tasks

[3] https://github.com/iesl/leopard

measure effectiveness of our method for CL. We include EWC (Kirkpatrick et al., 2017), which regularize the change of important model parameters during training. We also compare with LAMOL (Sun et al., 2019b), which learns to generate past training examples and replay them during training periodically.

Continual Meta-Learning Algorithms: Finally, we report the performance of continual meta-learning algorithms. We include MbPA++ (de Masson d'Autume et al., 2019) and meta-MbPA (Wang et al., 2020), which performs test-time adaptation over a few training examples stored in the memory. The latter includes a bi-level optimization objective that learns to adapt fast. However, both algorithms are only evaluated for adapting to seen training tasks only, with the goal of picking up performance on old tasks; we are the first to test their ability of adapting to new tasks.

3.2 Long-Short Term Hypernetwork with Regularization

We present a novel Long-Short Term Hypernetwork with Regularization (HNET+Reg.) to learn a fast adaptive model over a sequence of tasks while mitigating forgetting effects. The method is inspired by dynamic weight generation approaches for meta learning (Requeima et al., 2019; Rusu et al., 2019) and hypernetwork regularization for Continual Learning (Oswald et al., 2020). The method is illustrated in Figure 3, which consists of three components: (1) a context predictor to generate long-term or short-term task representations from training examples, and (2) a hypernetwork to generate weights of adapters (Houlsby et al., 2019) that can be plugged between the layers of pre-trained transformers, and (3) a regularization term to discourage change of adapters generated with past task embeddings. We discuss each individual component below.

Context Prediction. When the model receives an input $\boldsymbol{x}$ to predict the corresponding label, it is necessary to be known to which task the input data point belongs to. For this reason, we assume that each task can be characterized by a representation $\boldsymbol{z}$. The model should receive $\boldsymbol{z}$ to determine the context or task in addition to the input data points. We use a predictor $f_\phi(\boldsymbol{x}, \boldsymbol{y})$, implemented as a frozen BART encoder, to generate both long-term and short-term task representations, noted as $\boldsymbol{z}_\ell$ and $\boldsymbol{z}_s$ during training. The long-term representation

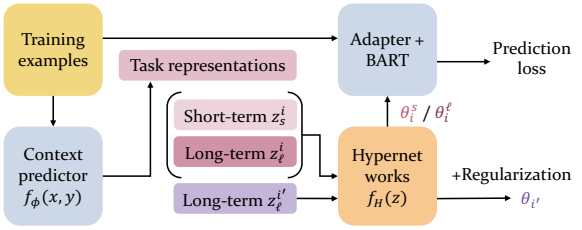

Figure 3: **Architecture for Long-Short Term Hypernetwork with Regularization.** A frozen context predictor f_ϕ generates long and short term task representations from training examples. The task representations are used to control a hypernetwork f_H to output weights of the adapters. The adapters are plugged into a frozen BART model to output predictions. We also feed the stored the long-term task representations $\boldsymbol{z}_\ell^{i'}$ of a previous task i' into the hypernetwork to generate adapter weights for solving previous tasks, and regularize its change to alleviate forgetting.

of task i, noted as $\boldsymbol{z}_\ell^i$, is the averaged BART representation of all examples in the training dataset of the task $\frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{x}_i \in \mathcal{D}_i} f_\phi(\boldsymbol{x}_i, \boldsymbol{y}_i)$, while the short-term representation $\boldsymbol{z}_s^i$ is averaged over the most recent N examples, $\frac{1}{N}\sum_{i=i'}^{i'-M+1} f_\phi(\boldsymbol{x}_i, \boldsymbol{y}_i)$. We store long-term task representation in a memory $\mathcal{M}$, which only introduces an overhead of one low-dimensional vector per training task.

As discussed in Sec. 2.1, the model may be evaluated on either seen or new tasks. To generate predictions on a seen task i, the model directly retrieves the stored task representation from the memory M. When used to perform few-shot learning over new tasks, the model generates task representations with f_ϕ over these training examples.

Adapter Generation with Hypernetworks. Our base model for prediction is a frozen BART model with adapters f_θ plugged after each layer of BART, including the final classification head over the entire vocabulary space. The adapted output of BART at layer ℓ is computed as $\boldsymbol{h}'_\ell = \boldsymbol{h}_\ell + f_\theta(\boldsymbol{h}_\ell)$, where $\boldsymbol{h}_\ell$ is the layer output before adaptation. We implement each adapter f_θ as a two-layer Multi-Layer Perceptron (MLP). A hypernetwork $f_H(\boldsymbol{z})$ takes as input the task representations $\boldsymbol{z}$ and generate the weights θ for each adapter.

Mitigating Catastrophic Forgetting: In our structure the PTLM is frozen and the learnable paramters of the prediction model are generated by the hypernetwork. Hence, forgetting effects mostly emerge from updates in the hypernetwork. Our approach to alleviate forgetting is adapted from Os-

wald et al. (2020), which proposed a CL algorithm that relies on hypernetworks. The idea is to regularize the change of generated model weights for previous tasks using the stored task representations. More specifically, before learning a new task $\mathcal{T}_i$, the hypernetwork generates adapter weights for each seen task $\mathcal{T}_{1..i-1}$, noted as $\boldsymbol{\theta}^i_{1..i-1}$. While updating the hypernetwork on the new task i, the model generate adapters for a random seen task from $1..i-1$ with stored low-dimensional long-term task representations, noted as $\boldsymbol{\theta}_{i'}$. It then penalizes the ℓ_2 distance between the adapter weights generated at the current time step and those generated before learning the new task, *i.e.*, $||\boldsymbol{\theta}_{i'} - \boldsymbol{\theta}^i_{i'}||^2_2$.

Advantage of HNET-Reg: (1) Unlike replay-memory based approaches (*e.g.*, MbPA (de Masson d'Autume et al., 2019)), HNET-Reg does not store any real training examples. It allows the method to be applied privacy-sensitive scenarios, where past data cannot be preserved; (2) Unlike existing methods that relies on test-time adaptation to recall performance on seen tasks (de Masson d'Autume et al., 2019; Wang et al., 2020), the approach predicts on seen tasks without adaptation, allowing more efficient inference.

Variants of HNET-Reg: As an ablation study, we experiment with HNET without the short term task representation (HNET$_{\setminus \text{ST}}$). We also report the performance of HNET with a significantly smaller hidden size (d=3 or 4) in the weight generator, so the total number of trainable parameters are smaller than training independent adapters.

3.3 Implementation Details

We use BART-base (Lewis et al., 2020) as our base pre-trained transformers. We use a learning rate of 1e-4 and a batch size of 64 in all experiments, except in adapter-only models, where we find 3e-4 performs better. 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 one previous task. In the few-shot learning stage, we use the same learning rate and train the model for 200 steps, assuming no validation sets to perform early stopping. 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 HNET is implemented as a two-layer MLP with a hidden size of 32 by default. 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. We note a method as a combination of model architecture and CL algorithm, *e.g.*, BART-Vanilla, HNET-EWC.

4 Results and Analysis

We address three research questions in this section. (1) How well can existing continual learning algorithms (including the proposed HNET) alleviate catastrophic forgetting? (2) Do existing algorithms enable a model to benefit from previous learned tasks for performance on unseen downstream tasks due to knowledge transfer? (3) Do resulting trained models with our approach show an improved performance on low-resource tasks?

4.1 Measuring Catastrophic Forgetting

Table 2 summarizes the experimental results. For continual learning methods, we report the performance at the end of training on the first row and the performance right after learning a task in the row below. The forgetting is measured by the drop of the performance at the end of training.

By comparing different continual learning methods, we notice vanilla online training clearly suffers from catastrophic forgetting, which archives non-trivial (above-majority guess) final averaged accuracy on only final three tasks. EWC could retain the performance on a early task SST-2 surprisingly well, but fails to perform well one other tasks. We notice that HNET could effectively mitigate forgetting, achieving non-trivial performance on 7 out of 9 tasks. We notice HNET$_{\setminus \text{ST}}$, which learns the model with long-term task-representation only, achieves better averaged final accuracy which is close to HNET trained offline. However, we will see later in the section that the short-term memory is crucial to the few-shot learning performance.

4.2 Measuring Knowledge Transfer

Next, we look into the performance right after learning a task in GLUE data stream in Table 3, compared to training individual models (HNET-Single and Adapter-Single in Table 2). We see continual learning over GLUE dataset overall improves the performance over single-task learning. We notice HNET-EWC achieves lower instant performance compared to HNET-Van, where the weight

Task	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	Average
Continual learning										
BART-Vanilla	0.00	0.00	0.00	0.00	0.00	4.04	49.46	52.71	43.66	20.68
BART-MbPA++	60.40	73.74	61.27	66.47	69.50	45.56	63.31	55.96	39.44	59.52
BART-meta-MbPA	30.87	70.41	68.38	49.53	67.06	32.75	76.50	62.09	43.66	55.69
HNET-Vanilla	0.00	49.54	31.37	49.53	63.18	0.00	63.92	70.76	56.34	42.75
HNET-EWC	0.00	80.16	0.00	0.00	0.00	0.00	51.18	60.65	57.74	27.76
HNET-Reg	30.87	89.90	63.23	82.13	81.04	75.55	75.03	62.09	60.56	68.93
HNET$_{\setminus ST}$-Reg	60.11	91.28	83.82	81.47	81.25	73.02	70.09	62.45	57.74	**73.47**
HNET$_{d=4}$-Reg	78.72	89.68	31.62	49.80	78.98	44.38	57.66	63.54	59.15	61.50
Single-task learning										
HNET-Single	78.52	90.25	85.54	85.00	82.31	75.77	86.40	53.43	56.34	**77.06**
Adapter-Single	69.13	91.28	77.94	84.20	82.53	75.19	85.50	52.71	56.34	74.98
Majority	69.13	50.92	68.38	50.47	63.18	35.33	50.54	52.71	56.34	55.22

Table 2: **Continual and offline learning performance on GLUE datasets evaluated at the end of the training**, where the number of training instances are limited to $10k$. The table header indicates the order of the tasks. Models may perform worse than majority guess because only exact match of generated answer sequences and the ground truth counts as a correct prediction.

Task	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	Average
BART-Vanilla	77.56	88.07	80.88	84.47	80.54	71.28	83.10	59.21	43.66	74.31
BART-MbPA++	80.15	91.97	80.64	84.73	83.25	75.50	85.56	58.84	56.34	77.48
BART-meta-MbPA	81.69	91.62	82.84	85.53	85.25	75.94	85.25	63.18	56.34	78.63
HNET-Vanilla	79.70	91.51	84.06	86.20	79.55	77.26	87.74	71.48	59.15	79.67
HNET-EWC	79.70	91.74	73.53	83.13	81.51	75.42	87.11	70.76	57.75	77.85
HNET-Reg	79.70	90.94	86.76	85.73	82.43	76.22	86.56	69.68	60.56	**79.84**
HNET$_{\setminus ST}$-Reg	78.04	90.82	86.52	86.00	81.95	75.52	87.06	62.45	57.74	78.46
HNET$_{d=4}$-Reg	79.19	92.43	70.09	83.20	79.13	75.83	86.40	70.40	59.15	74.01

Table 3: **Performance on GLUE datasets evaluated right after learning a task**. The table header indicates the order of the tasks.

regularization directly performed on adapters generators may have reduced its plasticity to generate adapters for new tasks. Overall, HNET-Reg achieves the best performance, implying the most positive knowledge transfer.

4.3 Measuring Few-shot Learning

Table 4 summarizes the results of learning on 17 few-shot learning tasks with 16 labeled examples for each class. We fine-tune the models that have continually learned on all glue tasks. We further include the performance of training individual adapters (BART-adapter) or the entire model (BART-single) for individual few-shot learning tasks, without pre-training on GLUE datasets. We notice HNET clearly improves the few-shot learning performance compared to single-task individual model baselines regardless of the measures taken to alleviate catastrophic forgetting, where HNET-Reg performs most competitively. We also notice the performance clearly drops when the short-term task representation is removed (HNET$_{\setminus ST}$). It aligns with our goal of incorporating short-term task representations, *i.e.,* allowing models to reliably generate adapter weights from only a small number of training examples.

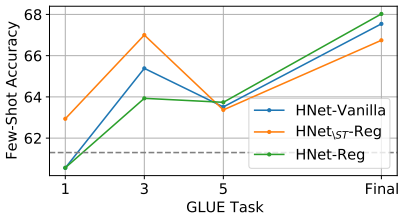

Figure 4: **Evolution of Few-shot learning performance over time**. We evaluate average few-shot learning accuracy over all 17 tasks at different time steps of GLUE task sequence training. The dash line shows the HNET-Single performance.

Figure 4 illustrate evolution of few shot learning performance over time. We measure averaged few-shot learning accuracy over all tasks as the model learns on the GLUE task sequence. The numbers shown are collected in a single run of experiments. We notice the few-shot learning performance generally improves as the model learn more GLUE tasks, which verifies the effect of learning to few-shot adapt and knowledge transfer. We further notice HNET-Reg starts to outperform other methods starting from the fifth task.

Task Task Num.	Entity Typing 2	Text Classification 10	NL Inference 1	Sentiment Analysis 4	Average 17
Continual learning					
BART-Vanilla	59.57	54.21	64.35	78.83	61.23
BART-MBPA	60.88	54.39	69.59	85.58	63.39
BART-meta-MbPA	63.26	54.14	71.93	84.80	63.47
HNET-Vanilla	67.40	58.75	78.46	85.83	67.30
HNET-EWC	66.54	58.19	69.04	84.73	66.06
HNET-Reg	66.60	58.78	73.75	87.88	**67.43**
HNET$_{\setminus ST}$-Reg	61.44	57.62	68.39	84.73	65.08
Single-task learning					
HNET-Single	64.71	56.70	57.90	71.95	61.30
BART-Adapter	61.76	54.32	58.61	71.87	59.58

Table 4: Few shot learning over 17 NLP tasks used in (Bansal et al., 2020) grouped by task category. We either uses the model obtained at the end of training on the GLUE task sequence, or train the model from initial BART.

4.4 Ablation Studies

Additional Diagnostic Dataset. To better compare with state-of-the-art continual learning algorithms in NLP (LAMOL (Sun et al., 2020), MbPA (de Masson d'Autume et al., 2019), meta-MbPA (Wang et al., 2020)), we incorporate a text classification data stream broadly used in these works, which consists of 5 text classification tasks (AGNews classification, Yelp sentiment, Amazon sentiment, DBPedia classification, Yahoo question and answers categorization) in 4 different orders (i-iv). The model visits each dataset in a single pass. We report the accuracy at end of training in Table 5; however, we note that the accuracy are not directly comparable as we use different model architectures. Nevertheless, from the comparison to the corresponding Multi-Task Learning performance, which is usually considered as an upper-bound of CL, we see HNET-Reg could most effectively alleviate forgetting.

Parameter Efficiency. We show the statistics of trainable and total parameters in each compared architecture in Table 6. In our default settings, HNET 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. As shown in Table 2, HNET$_{d=4}$-Reg achieves a final accuracy of 61.50, which is still higher than BART-MBPA and BART-meta-MBPA. On five text classification tasks, as shown in Table 5, HNET$_{d=3}$-Reg could also effectively reduce catastrophic forgetting, with a performance gap of only 1.7 compared to the multi-task learning upper bound.

Order	MbPA	Meta-MbPA	LAMOL	HNET-Reg	HNET$_{d=3}$-Reg
i.	75.3	77.9	76.7	75.1	73.4
ii.	74.6	76.7	77.2	73.8	73.6
iii.	75.6	77.3	76.1	74.8	68.3
iv.	75.5	77.6	76.1	73.9	73.7
Average	75.3	77.3	76.5	74.4	72.7
MTL Avg.	78.9	78.9	78.9	74.4	74.4
Diff.	-3.6	-1.6	-2.4	0.0	-1.7

Table 5: Continual and multi-task learning (MTL) performance over four text classification tasks over four different orders. The results of MbPA++, Meta-MbPA, and LAMOL are cited from (Wang et al., 2020) and use a different model architecture, and thus numbers are not directly comparable. We also show differences between the corresponding MTL and CL performance.

5 Related Work

Our work lies on the intersection of continual learning and meta-learning.

5.1 Continual Learning

The primary challenge that is addressed in CL literature is overcoming tackle catastrophic forgetting. Existing CL methods use experience replay or model regularization for this purpose. The idea behind experience replay is replaying representative samples of past tasks through the pseudo-rehearsal process (French, 1999; Robins, 1995). The representative samples are stored in a memory buffer and should be selected such that they contribute significantly in learning the corresponding task. Several strategies have been developed to select these samples. An effective strategy is select samples that introduce strong gradient descent signal during model training (Schaul et al., 2016). The idea of model regularization (Kirkpatrick et al., 2017) is to constrained the network when a new task is learned. For examples, if we consolidate weights that are important to perform well on a given task after learn-

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

Table 6: Statistics of trainable and total model parameters in each model to learn 9 GLUE tasks.

ing it, new tasks will be learned through plastic weights, leading to preserving knowledge on past tasks. An alternative strategy is rather constraining the network, we expand the network progressively to learn the new tasks using new weights (Rusu et al., 2016). The idea that we explored in this work is based on regularizing a hypernetwork that generates weights for the downsream prediction model (Oswald et al., 2020).

5.2 Meta-Learning

The primary goal in meta-learning is to benefit from cross-task knowledge transfer across related tasks to train a shared model more efficiently in terms of required per task samples for learning (Schmidhuber, 1987; Finn et al., 2017b). Despite significant difference, meta-learning algorithms recasts meta-learning as a task inference algorithm, where the goal is to update the downstream model to work within the input task context. The pioneer work by Finn et al. (Finn et al., 2017b) infers the task context using a few data points that are used to update the model using only a few gradient descent steps. CAVIA (Zintgraf et al., 2019) model the context as an independent variable in the model which is inferred from the few input data points using a few gradient descent steps. The general approach that we explored, addresses meta-learning using a hierarchical models, where a hyper-learner model generates task-specific parameters to adapt the task-specific model (He et al., 2019). In most meta-learning works, the assumption is that all the tasks are accessible simultaneously similar to multitask learning. When the tasks are learned sequentially, meta-learning methods are vulnerable with respect to catastrophic forgetting. As we explored, forgetting can be addressed in the hierarchical formulation of meta-learning by tackling forgetting in the hypernetwork (Oswald et al., 2020).

6 Conclusion and Discussion

We proposed an algorithm for learning a set of sequentially arriving NLP tasks. Our algorithm is based on using a hypernetwork to update an adapter network to perform well on the input task. We benefit from model regularization to tackle catastrophic forgetting in the hypernetwork. Our method also enables the model to infer the input task given only a few labeled input data points and then adapt the model accordingly. We demonstrate that our approach not only addresses catastrophic forgetting but enables the model to benefit from knowledge transfer to improve its performance on future tasks. Future work includes extending our work to one-shot model adaptation setting for a fully task-agnostic approach.

References

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

Magdalena Biesialska, Katarzyna Biesialska, and Marta R Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6523–6541.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chelsea Finn, P. Abbeel, and Sergey Levine. 2017a. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017b. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

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

X. He, Jakub Sygnowski, Alexandre Galashov, Andrei A. Rusu, Y. Teh, and Razvan Pascanu. 2019. Task agnostic continual learning via meta learning. *ArXiv*, abs/1906.05201.

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*.

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*.

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.

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.

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*.

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

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

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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.

A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. 2016. Progressive neural networks. *ArXiv*, abs/1606.04671.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2019. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. In *IJCLR*.

Jürgen Schmidhuber. 1987. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. Ph.D. thesis, Technische Universität München.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019a. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019b. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

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

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

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

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.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. 2019. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702. PMLR.