HOP to the Next Tasks and Domains for Continual Learning in NLP

Umberto Michieli, Mete Ozay

Samsung Research UK
{u.michieli, m.ozay}@samsung.com

Abstract

Continual Learning (CL) aims to learn a sequence of problems (*i.e.*, tasks and domains) by transferring knowledge acquired on previous problems, whilst avoiding forgetting of past ones. Different from previous approaches which focused on CL for one NLP task or domain in a specific use-case, in this paper, we address a more general CL setting to learn from a sequence of problems in a unique framework. Our method, HOP, permits to *hop* across tasks and domains by addressing the CL problem along three directions: (i) we employ a set of adapters to generalize a large pre-trained model to unseen problems, (ii) we compute high-order moments over the distribution of embedded representations to distinguish independent and correlated statistics across different tasks and domains, (iii) we process this enriched information with auxiliary heads specialized for each end problem. Extensive experimental campaign on 4 NLP applications, 5 benchmarks and 2 CL setups demonstrates the effectiveness of our HOP.

Introduction

Current practice to obtain a deep learning model to perform a specific assignment is to train the model on a specific dataset for that particular assignment (Chen and Liu 2018). However, this paradigm has several inherent limitations. For example, as models get larger, training from scratch requires a larger amount of expensive labeled data and computation time, which can be reduced by knowledge transfer (KT) from a pre-trained model on a base domain. This problem is generally addressed via adapting (*e.g.*, through fine-tuning, FT) large pre-trained language models (*e.g.*, BERT, Devlin et al. 2019) to various downstream NLP applications, such as Text Classification (TC, Wang et al. 2023a; Wu et al. 2023; Ke, Xu, and Liu 2021; Hu et al. 2021), Natural Language Inference (NLI, Pfeiffer et al. 2020), Document (Ke et al. 2020) or Aspect (Ke, Xu, and Liu 2021; Zhou et al. 2021) Sentiment Classification (DSC and ASC). In our work, we employ the Adapter-BERT (Houlsby et al. 2019) model. FT a large pre-trained model reaches state-of-the-art results on NLP benchmarks with a *static* distribution. However, if a stream of problems[1] are presented sequentially, the naïvely FT model faces catastrophic forgetting (CF, McCloskey and Cohen 1989) of previous knowledge due to the non-stationary data distribution, and cannot make use of past knowledge to improve capability on subsequent problems (forward KT, Lopez-Paz and Ranzato 2017) or vice-versa (backward KT). In particular, high CF and low KT hinder performance of CL for NLP, as several NLP applications share similar knowledge that can be exploited to achieve higher accuracy on future/previous problems, without accuracy degradation. Indeed, ideally, learning a sequence of problems should allow multiple problems to support each other via KT (Rusu et al. 2016; Ke, Liu, and Huang 2020). To address such issues towards more versatile NLP models, we focus on CL to tune a pre-trained Adapter-BERT for a stream of problems.

In CL, a model learns a sequence of problems incrementally. After each incremental learning (IL) stage is completed, its training data is typically discarded (Chen and Liu 2018). Three main families of CL setups can be identified (Van de Ven and Tolias 2019): namely, Task-IL (TIL), Domain-IL (DIL), and Class-IL (CIL). TIL builds one model for each task (*e.g.*, to classify sentiment in products' reviews). At test time, task identifier specifies the proper model for each input sample. This could significantly increase the number of parameters; however, in our case, most of the parameters are shared across problems and only a few parameters are problem-specific. DIL is similar to TIL, however, builds a single head for each domain as classes are shared across domains. In DIL, no identifier is required at test time and subsequent problems present data from different domains (*e.g.*, reviews from online commerce, or from movie critique, *etc.*). In CIL, non-overlapping classes are learned progressively. Opposed from traditional CL approaches used in Computer Vision (CV), most of the NLP problems are formulated as either TIL or DIL (Biesialska, Biesialska, and Costa-jussà 2020; Ke and Liu 2022; Ke et al. 2021b; Sun, Ho, and Lee 2019) and, to the best of our knowledge, no prior work has addressed them both jointly.

Differently from concurrent CL NLP approaches, in this paper, we evaluate models on both TIL and DIL in a unified framework which employs parameter-efficient transfer learning strategies to adapt the models to each end problem: (i) as in current state-of-the-art approaches (Ke, Xu, and Liu 2021; Ke, Liu, and Huang 2020), we use Adapter-BERT

[1] For the sake of clarity, we refer to *problems* as either *tasks* or *domains* experienced by the CL method over time.

with a separate set of adapters tuned for each problem; (ii) we discard the [CLS] token, which we show being counterproductive for CL, and rather compute high-order statistical measures over the distribution of extracted features (*i.e.*, tokens representing text embeddings); (iii) we use an MLP head specialized for each problem to process and combine such information. With a slight abuse of notation, we call our method HOP, from High-Order Pooling, that is the most distinctive component of our method.

Our HOP extracts multiple cues from the limited samples drawn from non-stationary distributions while preserving previous knowledge. HOP accurately models the variable distribution of problems since input-level distribution shift is reflected into feature-level distribution shift. Indeed, we show that this variation is not properly captured by the single [CLS] token. We present an extensive validation on 2 CL setups (DIL and TIL) and 5 benchmarks outperforming current state-of-the-art on accuracy, KT, CF, and runtime.

Related Work

For a wide survey of the current state-of-the-art, we refer to recent CL reviews (De Lange et al. 2021; Lesort et al. 2020; Michieli, Toldo, and Zanuttigh 2022; Biesialska, Biesialska, and Costa-jussà 2020; Ke and Liu 2022).

Traditional CL methods focused on image classification and can be grouped according to the proposed technique. (1) Regularization-based methods are generally based on knowledge distillation (Li and Hoiem 2017; Jung et al. 2016; Michieli and Zanuttigh 2019) or on importance score for each parameter to compute a penalty term in the optimization to reduce weight deviation while learning new problems (Kirkpatrick et al. 2017; Zeng et al. 2019; Nguyen et al. 2018; Zenke, Poole, and Ganguli 2017; Ahn et al. 2019). (2) Parameter-isolation approaches dedicate a set of parameters to each problem to reduce forgetting when learning subsequent problems. Parameters can be either masked out (Serra et al. 2018; Mallya and Lazebnik 2018; Mallya, Davis, and Lazebnik 2018; Wang et al. 2023b), frozen (Rusu et al. 2016; Xu and Zhu 2018; Michieli and Zanuttigh 2021), or new branches are grown over time (Rusu et al. 2016; Xu and Zhu 2018). (3) Replay-based methods either retain an exemplar set of previously seen data (Rebuffi et al. 2017; Isele and Cosgun 2018; Lopez-Paz and Ranzato 2017; Chaudhry et al. 2018) or generated pseudo-samples (Shin et al. 2017; Maracani et al. 2021) to reduce CF and promote KT.

CL in NLP is in rapid expansion due to its great importance. Early works tackled lifelong learning (with no knowledge preservation, hence no CF) for sentiment analysis (Carlson et al. 2010; Silver, Yang, and Li 2013; Ruvolo and Eaton 2013; Chen, Ma, and Liu 2015; Wang et al. 2019; Qin, Hu, and Liu 2020; Wang et al. 2018). Recent works have dealt with CF in many applications: sentiment analysis (Lv et al. 2019; Ke et al. 2021b; Ke, Xu, and Liu 2021), dialogue systems (Shen, Zeng, and Jin 2019; Madotto et al. 2020; Qian, Wei, and Yu 2021; Chien and Chen 2021), language modeling (Sun, Ho, and Lee 2019; Chuang, Su, and Chen 2020) and learning (Li et al. 2019), cross-lingual modeling (Liu et al. 2020), sentence embedding (Liu, Ungar, and Sedoc 2019), machine translation (Khayrallah et al. 2018; Zhan et al. 2021), question answering (Greco et al. 2019), named entity recognition (Monaikul et al. 2021).

Most of the previous literature focuses on the simpler TIL setup. SRK (Lv et al. 2019) and KAN (Ke et al. 2020) tackled DSC via recurrent architectures. They are mainly conceived for KT, hence they suffer from CF and cannot be easily extended to BERT. B-CL (Ke, Xu, and Liu 2021) is the first CL framework for ASC: it employs Adapter-BERT and is based on capsule network and dynamic routing, bringing only limited KT. CAT (Ke, Liu, and Huang 2020) works on mixed sequences of similar and dissimilar problems, and can transfer knowledge among similar problems. Snapshot (Wang et al. 2023a) regularizes training with adapters learned over previous problems. Parallel to CL, AdapterFusion (Pfeiffer et al. 2021) uses a two-stage method to learn the adapters of Adapter-BERT to improve multi-task learning, hence it has no CF. CTR (Ke et al. 2021a) extends the adapters concept to the idea of CL plugins to adapt BERT to each problem, and it is the state-of-the-art in TIL.

Recently, DIL has gained attention. LAMOL (Sun, Ho, and Lee 2019) uses a language model (*i.e.*, GPT-2 Radford et al. 2019) to solve sequential problems and to generate training pseudo-samples against CF; CLASSIC (Ke et al. 2021b) uses contrastive learning to promote KT.

Pooling in NLP has been recently studied to improve accuracy (Wu et al. 2020; Ács, Kádár, and Kornai 2021; Zhao et al. 2022). In particular, Wu et al. 2020 propose an attentive pooling scheme with learnable norm to extract accurate text representations in different problems, motivated by 3 observations: (i) different contexts have different informativeness for learning text representations (*e.g.*, they might be important to determine sentiment polarity, however, probably less relevant for TC); (ii) different problems have different characteristics; (iii) popular pooling methods (*e.g.*, MAX or AVG) may over-emphasize some concepts and disregard other useful contextual information. To summarize, some problem specific words or sentences contain information regarding output class in various ways. Our work is motivated by these results. However, such pooling schemes cannot be applied to CL. To cope with this, our HOP computes multiple statistical moments from the encoded text to capture evolution of different statistics of the input.

Problem Formulation

CL learns a sequence of problems $t \in \{1, \dots, T\}$. Each problem t has its test data $\hat{\mathcal{S}}_t$ and training data $\mathcal{S}_t = \{(x_t^k, y_t^k)\}_{k=1}^{N_t 0}$, where $x_t^k \in \mathcal{X}_t$ is a training sample with label $y_t^k \in \mathcal{Y}_t$ (*i.e.*, supervised problems). Then, the CL goal is to minimize the empirical loss $\mathcal{L}$ over all seen problems. At problem T, we aim at training models $f_t, \forall t$, parameterized by θ (*i.e.*, $\hat{y}_t^k = f_t(x_t^k; \theta)$), which minimize the loss

$$\sum_{t=1}^{T} l_t, \quad \text{with} \;\; l_t = \frac{1}{N_t} \sum_{k=1}^{N_t} \mathcal{L}(\hat{y}_t^k, y_t^k). \tag{1}$$

However, Eq. (1) cannot be minimized since no (in case of replay-free CL methods) or limited (in case of replay CL

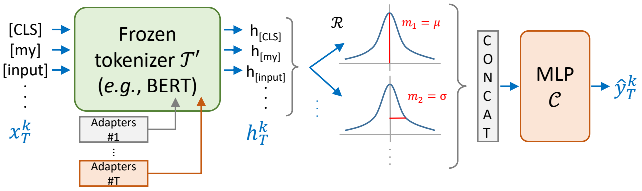
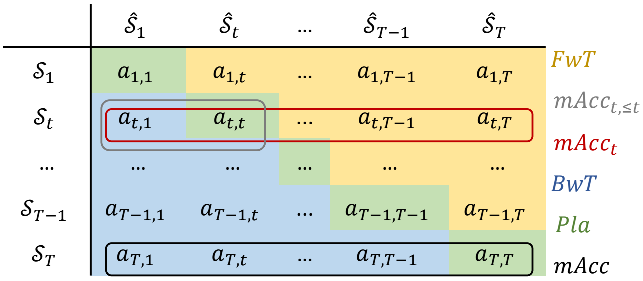


Figure 1: The proposed HOP framework. During the incremental step T, only orange modules are trained, while gray and green modules are frozen.



	$\hat{\mathcal{S}}_1$	$\hat{\mathcal{S}}_t$	...	$\hat{\mathcal{S}}_{T-1}$	$\hat{\mathcal{S}}_T$
$\mathcal{S}_1$	$a_{1,1}$	$a_{1,t}$	...	$a_{1,T-1}$	$a_{1,T}$
$\mathcal{S}_t$	$a_{t,1}$	$a_{t,t}$	...	$a_{t,T-1}$	$a_{t,T}$
...	...	...	...	...	...
$\mathcal{S}_{T-1}$	$a_{T-1,1}$	$a_{T-1,t}$	...	$a_{T-1,T-1}$	$a_{T-1,T}$
$\mathcal{S}_T$	$a_{T,1}$	$a_{T,t}$	...	$a_{T,T-1}$	$a_{T,T}$

FwT
$mAcc_{t,\leq t}$
$mAcc_t$
BwT
Pla
$mAcc$

Figure 2: Accuracy matrix showing the main CL metrics used in this work. $\hat{\mathcal{S}}_t$ and $\mathcal{S}_t$ are the testing and training datasets at step t.

methods) access to previous data is guaranteed. In the most challenging replay-free setup, we can minimize the empirical loss on the current problem T only, *i.e.*, l_T.

Therefore, CL methods try to approximate Eq. (1) in different ways (*e.g.*, via regularization, replay, *etc.*). Instead, we extract high-order statistics from the input dataset and we process this additional information via an auxiliary problem-specific multi-layer perceptron (MLP) to adapt the current model to the current problem.

Depending on the properties of $(\mathcal{X}_t, \mathcal{Y}_t)\ \forall t$, we can identify TIL, DIL and CIL. In CIL, models are progressively trained with new classes, and it has been less attractive for NLP applications as the number of classes is generally determined *a priori* (Ke et al. 2021b). Therefore, we address both TIL and DIL in a unified framework of CL for NLP.

Our Proposed Framework: HOP

Our framework employs parameter-efficient learning strategies to adapt models to each end problem, namely: (i) adapter modules, (ii) computation of high-order moments, and (iii) a specialized MLP for each end problem. In the experimental validation, we apply HOP to BERT-based models due to its superior performance in NLP, however, it can be seamlessly applied to other architectures as well.

(i) Following recent CL approaches for NLP (Ke, Xu, and Liu 2021; Ke, Liu, and Huang 2020), our best architecture relies on Adapter-BERT, with a separate set of adapters tuned for each problem. Adding adapters to BERT is a highly parameter-efficient transfer learning paradigm: in CL, this means that subsequent problems have separate adapters (which are small in size). An adapter layer is a tunable 2-layer fully-connected network, which adapts the pretrained model to the end problem at hand. In this way, there is no need for a separate BERT model fine-tuned on each problem, which is extremely parameter-inefficient if many problems are learned in sequence.

(ii) HOP computes high-order statistical moments from the distribution of extracted tokens to capture most of the information from the input sequence. Current approaches (Ke, Xu, and Liu 2021; Ke et al. 2021a) design their CL systems relying on the `[CLS]` token embeddings. More formally, we consider f_t composed of a tokenizer $\mathcal{T}$ and a classifier $\mathcal{C}$ by $f_t = \mathcal{C} \circ \mathcal{T}$ to recognize N_C classes ($\circ$ represents function composition). Traditionally, $\mathcal{T}$ includes a reduction function $\mathcal{R}$ as the last layer to summarize the whole input sequence into one element. Therefore, we can write $\mathcal{T} = \mathcal{R} \circ \mathcal{T}'$, with $\mathcal{T}'$ being the tokenizer without the final reduction function. $\mathcal{R}$ is often identified by the `[CLS]` token or by AVG pooling. However, different problems usually have different peculiar patterns in the input samples, and the output should be an explicit function of the whole, non-reduced, embedding sequence. Therefore, we compute high-order *central* moments from the input sequence and concatenate (concat) them. We define the reduction function by

$$\mathcal{R} = \mathrm{concat}(m_1, m_2, \ldots, m_p), \tag{2}$$

where p is the order of considered moments, that is, m_1 is the first moment (*i.e.*, AVG), m_2 is the second moment (*i.e.*, the variance), *etc.* (Papoulis and U. Pillai 2002; Michieli, Parada, and Ozay 2023; Michieli and Ozay 2023; Michieli et al. 2024). Such moments are computed over the distribution of tokens identified by the unreduced tokenizer $\mathcal{T}' : x_t^k \mapsto h_{t,d}^k$ where d denotes the dimension of the embedded sequence and each $h_{t,d}^k \in \mathbb{R}^Q$ with Q channels. This step may recall the statistical moments of tokens identified by standard N-gram models (Cortes and Mohri 2004).

(iii) We process and combine embeddings computed by $\mathcal{T}$ with an auxiliary MLP head specialized for each problem. We replace the usual linear layer constituting $\mathcal{C}$ with an MLP. The MLP head increases the adaptation capacity to process the high order information while being highly parameter efficient. We design the MLP as a 2-layer network consisting of $p \cdot Q$ and N_C neurons at each layer, respectively.

Overall, our HOP can extract richer information from the samples drawn from the non-stationary input sequence distributions while preserving previous knowledge. Therefore, our method can *hop* across the distributions of subsequent tasks and domains, since input-level distribution shift is reflected into a feature-level distribution shift via the embedding tokenizer. Our framework is applicable to both TIL and DIL setups: in TIL, one head per task adapts models to each of the tasks separately; in DIL, one single head is incrementally adapted to the varying domains.

HOP alleviates CF thanks to (i) frozen shared backbone to extract shared rich features, and (ii) adapters tuned for each problem. It promotes KT via (i) initialization of adapters to the last achieved ones, therefore bringing onward the previous information; (ii) modeling tailed distributions of tokens encountered in subsequent CL problems with different moments. Finally, MLP heads improve plasticity for new tasks.

An overview of our framework is given in Figure 1, where we depict the modular property of HOP, so it can be plugged on top of other CL methods. Compared to competing methods, HOP only brings a minimal computation and complexity footprint. Detailed analyses are provided in Sec. .

The main novelty of our HOP are: (i) we are the first to address tailed distribution of features and its relation to CL; (ii) we revisit high-order feature statistics (inspired from N-grams co-occurrence statistics) to deep NLP architectures; (iii) we exploit them in CL for the first time; (iv) we use adapters for CL extending concurrent works to jointly tackle TIL/DIL; (v) we include specialized MLP heads to improve plasticity in learning new concepts. Overall, we propose a simple yet effective CL baseline that outperforms complex architectures in all experiments, while requiring significantly lower training time. Additionally, it can be seamlessly applied on top of competitors.

Experimental Setup

Architectures. We evaluate HOP on 3 BERT-based architectures (Devlin et al. 2019). Previous works (Ke et al. 2021a; Ke, Xu, and Liu 2021) have shown that naïvely fine-tuning BERT increases CF, hence we focus on more versatile ways to learn new concepts limiting CF. *BF + Lin* consists in a Frozen BERT (BF) with a trainable linear layer on top. *BF + CNN* consists in a BF with a trainable CNN TC network (Kim 2014) on top. *Adapter-BERT* trains only the adapter blocks built of 2 linear layers each with 2000 neurons.

Datasets. We consider 4 applications, unifying the setups proposed by (Ke et al. 2021b,a; Asghar et al. 2020). (1) ASC classifies a review sentence on either *positive*, *negative* or *neutral* aspect-level sentiments. We use 19 datasets (*i.e.*, reviews of 19 products) taken from four sources: 5 products from HL5Domains (Hu and Liu 2004), 3 products from Liu3Domains (Liu et al. 2015), 9 products from Ding9Domains (Ding, Liu, and Yu 2008), and 2 products from SemEval14 Task 4 (Pontiki et al. 2014). We applied the same data filtering of previous works for fair comparison (Ke et al. 2021a). (2) DSC classifies product reviews into either *positive* or *negative* opinion classes, using TC formulation of Devlin et al. 2019. We use 10 DSC datasets (*i.e.*, reviews of 10 products, Ke et al. 2020). We consider both a small training version of 100 positive and 100 negative reviews per problem, and the full training version of 2500 positive and 2500 negative reviews per problem. Validation and test sets are fixed, and each consists of 250 reviews per each class. The first experiment is arguably more useful in practice because labeling a large number of examples is costly, thus, ablation is carried out on this split. (3) TC classifies text into 20 classes using 20News data (Lang 1995). We split documents into 10 problems with 2 classes per problem (in DIL, N_C is supposed known *a priori*). Classes are variegate and share little knowledge, hence forgetting is the main issue. (4) We target NLI for sentence understanding using the MultiNLI dataset (Williams, Nangia, and Bowman 2018) which is one of the largest corpus of its kind. Sentences are classified into: *entailment*, *neutral* and *contradiction*. We split data in 5 problems, each belonging to a specific domain (*fiction*, *telephone*, *etc*., Asghar et al. 2020).

Baselines. As the first baseline, we consider a separate model learned for each problem independently, which we call SDL (standalone). No KT or CF occurs here. Second, we compare against FT which simply optimizes the model over the sequence of problems. Third, we examine 13 CL competitors. Among them, some approaches have been proposed for CL in NLP and have been briefly described in Sec. . Additionally, we adapted CL methods proposed in CV for our NLP applications. For TIL-based works: UCL (Ahn et al. 2019) proposes uncertainty-based regularization via a Bayesian online learning framework; HAT (Serra et al. 2018) focuses on employing embeddings preserving information of previous problems while learning new ones. CIL approaches can be adapted to TIL by training the head of the specific problem and considering predictions of the specific head during testing. Among them, we used (i) regularization-based methods, *e.g.*, EWC (Kirkpatrick et al. 2017), OWM (Zeng et al. 2019), and L2 (Kirkpatrick et al. 2017), and (ii) replay-based methods, *e.g.*, the efficient A-GEM (Chaudhry et al. 2018), and DER++ (Buzzega et al. 2020) for pseudo replay. We note that a single upper bound for accuracy does not exist, since different methods have a different number of parameters, as shown in Table 3.

Hyperparameters. We employ the same scenarios as current state-of-the-art approaches (Ke et al. 2021a). We follow the CL evaluation of (De Lange et al. 2021): after training on one problem is completed, the respective training data is no longer accessible; all hyperparameters are chosen according to the performance on the validation set; after all problems are learned, testing is carried out on the test set. We report results averaged over 5 random seeds (*i.e.*, different ordering of problems) and we report the mean (the standard deviation is negligible - lower than 0.1 in all cases). All baseline approaches use the `[CLS]` token as the reduction function of the tokenizer. We show that this is a major limitation, concurrently hinted also by (Mirzadeh et al. 2022). However, (Mirzadeh et al. 2022) address the issue only marginally, while we propose a simple and effective framework to overcome it. The only hyperparameter specific to our framework is p, which is set according to the best validation results. Empirically, $p = 3$ provides the best results and represents a good compromise with additional computational complexity.

Metrics. We compute both mean accuracy (mAcc, ↑) and macro-F1 (MF1, ↑), to reduce biases in accuracy originating from imbalanced classes. To fully characterize the different approaches, we report a wide range of forgetting and transfer metrics computed from the accuracy matrix (Mai et al. 2022), as illustrated in Figure 2. Namely, we report: backward transfer (BwT, ↑), which tracks the influence that learning a new problem has on the preceding problems performance, to measure stability; forward transfer (FwT, ↑) measures the positive influence of learning a problem on future problems performance; forgetting ($Forg$, ↓) averages the difference of class-wise accuracy achieved at the last step and the best class-wise accuracy achieved previously; plasticity (Pla, ↑) averages the accuracy achieved on each problem evaluated right after learning that problem. Additionally, we report the number of overall (#OP, ↓) and of

	Method	**ASC**				**DSC (small)**				**DSC (full)**				**20News**				**NLI**			
		TIL		**DIL**		**TIL**		**DIL**		**TIL**		**DIL**		**TIL**		**DIL**		**TIL**		**DIL**	
		mAcc	MF1	mAcc	MF1	mAcc	MF1	mAcc	MF1	mAcc	MF1	mAcc	MF1	mAcc	MF1	mAcc	MF1	mAcc	MF1	mAcc	MF1
BF+Lin	SDL	56.32	27.8	58.48	35.21	48.97	42.21	57.87	56.97	82.32	80.11	77.45	76.94	95.31	95.30	54.96	54.31	48.69	43.78	43.24	41.55
	FT	61.18	32.00	68.42	49.97	60.41	52.36	66.12	65.06	70.53	63.62	75.07	74.19	66.30	65.59	55.85	54.88	42.53	37.09	41.08	38.17
	HOP	**86.38**	**78.64**	**87.54**	**81.79**	**78.48**	**75.31**	**81.96**	**80.93**	**84.59**	**83.18**	**85.63**	**84.65**	**96.45**	**96.45**	**68.45**	**66.16**	**61.05**	**60.88**	**59.24**	**58.82**
Bert (frozen) + CNN	SDL	78.14	58.13	78.14	58.13	73.88	67.97	73.88	67.97	85.34	80.17	85.34	80.17	96.49	96.48	96.49	96.48	68.21	67.49	68.21	67.49
	FT	85.51	76.64	86.85	78.73	83.12	79.23	85.66	84.87	61.88	45.79	85.54	84.59	83.28	81.81	64.45	58.68	71.98	71.70	71.70	71.47
	L2	56.04	38.40	86.31	77.90	59.17	48.39	82.29	81.32	69.80	62.63	64.92	61.06	72.14	65.39	62.50	58.85	56.80	56.49	68.71	68.55
	A-GEM	86.06	78.44	79.74	71.78	59.33	45.94	80.23	79.27	70.67	61.77	87.53	86.61	93.31	92.95	58.50	46.86	71.95	71.89	72.49	72.34
	DER++	84.27	75.08	87.53	80.09	72.29	66.28	83.26	82.29	86.70	85.46	86.87	85.99	60.44	49.67	54.20	39.22	70.48	69.98	71.84	71.69
	EWC	86.37	74.52	86.60	78.31	82.38	78.41	81.88	80.99	72.77	65.76	85.78	84.65	80.26	78.60	**75.40**	71.98	68.85	68.18	71.10	70.90
	OWM	87.02	79.31	86.11	76.65	58.07	42.63	80.25	79.45	86.30	85.36	82.31	81.44	84.54	82.73	50.00	33.33	72.23	71.96	65.97	65.81
	UCL	83.89	74.82	85.38	76.90	80.12	74.13	84.39	83.59	74.76	69.48	89.00	88.27	94.65	94.63	72.20	67.15	71.61	71.46	72.66	72.50
	HAT	86.74	78.16	84.73	76.49	79.48	72.78	79.98	79.10	87.29	86.14	88.07	87.26	93.51	92.93	53.80	42.56	69.78	69.70	71.06	70.93
	CAT	83.68	68.64	87.32	81.42	67.41	56.22	71.03	69.40	87.34	86.51	83.56	82.74	95.17	95.16	51.15	44.11	72.39	72.08	73.21	73.10
	KAN	85.49	77.38	83.20	73.52	77.27	72.34	68.02	66.13	82.32	81.23	86.10	85.30	73.07	69.97	64.95	58.84	72.77	72.72	73.97	73.86
	SRK	84.76	78.52	83.91	74.38	78.58	76.03	75.58	74.64	83.99	82.66	88.14	87.45	79.64	77.89	57.80	51.26	69.84	69.79	70.69	70.54
	HOP	**87.51**	**80.45**	**88.16**	**82.76**	**83.79**	**81.45**	**85.74**	**84.98**	**87.98**	**86.91**	**89.30**	**88.54**	**95.23**	**95.20**	74.32	**72.10**	**74.49**	**74.34**	**76.59**	**76.45**
Adapter-BERT	SDL	85.96	78.07	85.96	78.07	76.31	71.04	76.31	71.04	88.30	87.31	88.30	87.31	96.20	96.19	96.20	96.19	72.54	71.26	72.54	71.26
	FT	54.03	44.81	86.67	78.04	55.19	35.28	82.22	81.32	64.94	63.40	88.77	88.21	68.29	61.70	63.10	57.87	69.31	67.80	81.63	81.59
	L2	63.97	52.43	75.64	59.78	70.87	69.11	83.78	82.86	73.03	71.50	89.02	88.49	69.56	65.50	63.85	61.29	77.16	76.85	78.81	78.91
	A-GEM	45.88	28.21	86.78	77.84	59.35	54.20	86.25	85.49	71.22	69.94	86.54	56.12	60.29	50.40	62.24	60.30	78.84	78.04	77.45	77.36
	DER++	47.63	35.54	88.59	79.85	63.11	61.96	84.00	83.16	59.67	57.82	87.12	86.43	58.95	49.58	60.18	57.89	74.12	73.78	79.84	79.56
	EWC	56.30	49.58	88.05	78.75	58.34	42.85	87.29	86.34	62.69	61.51	88.58	87.94	61.86	53.94	49.95	36.86	75.72	75.38	80.63	80.59
	OWM	72.99	66.51	87.66	78.82	73.97	71.96	77.32	76.68	85.46	84.57	85.78	85.45	71.10	66.25	60.23	57.94	70.45	68.24	77.65	76.89
	UCL	64.46	36.64	71.23	39.61	48.36	32.07	56.53	52.96	57.06	55.86	88.18	87.54	51.75	36.06	50.25	36.78	76.54	76.43	80.36	80.40
	HAT	86.14	78.52	88.23	79.19	80.83	78.41	86.22	85.44	88.00	87.26	86.52	85.78	95.22	95.21	63.30	60.44	71.51	68.83	81.02	80.92
	B-CL	88.29	81.40	89.83	**84.22**	84.34	83.12	85.92	85.11	79.76	76.51	88.12	87.48	95.07	95.04	64.50	61.87	72.92	72.71	81.23	81.02
	CTR	89.47	83.62	89.13	83.52	83.96	83.00	86.01	85.16	89.31	88.75	88.36	87.89	95.25	95.23	65.76	63.04	75.95	75.46	80.78	80.64
	HOP	**89.84**	**85.06**	**89.87**	84.12	**85.63**	**84.18**	**87.84**	**86.96**	**90.08**	**89.44**	**89.93**	**89.41**	**95.30**	**95.29**	**72.30**	**69.67**	**81.75**	**81.66**	**82.46**	**82.22**

Table 1: $mAcc$ and $MF1$ over 5 benchmark datasets on both TIL and DIL setups. We evaluate 3 network architectures based on BERT and 14 baselines. Results are color-coded according to the column-wise value and best results are in bold.

trainable parameters (#TP, ↓), and the computation time (↓, in minutes).

Experimental Results

Main Results. We evaluate methods on 5 benchmarks (ASC, DSC small, DSC full, 20News, NLI) targeting 4 applications (ASC, DSC, TC, NLI) in 2 CL setups (DIL and TIL) using 3 network architectures based on BERT. Table 1 shows that HOP clearly outperforms or achieves comparable results to baseline competitors in every scenario.

In the first block, we evaluate them on BF + Lin. Due to the low accuracy of this architecture, we compare our framework only against SDL and FT. HOP outperforms both by a large margin in every case. In the second block, we use BF + CNN (Kim 2014). Here, we report comparison against several approaches (we note that B-CL and CTR cannot be employed with a CNN head). Finally, the best results are achieved on Adapter-BERT reported in the third block (we remark that CAT, KAN, SRK cannot work with adapters). We observe that $mAcc$ and $MF1$ generally show consensus in identifying the best methods. Also, results are higher in DIL since a single head can transfer knowledge more easily.

SDL outperforms some approaches, due to increased model size for adaptation to end problems. However, it builds a model for each problem independently using a separate network, therefore, it does not handle CF or KT. On the other hand: FT, regularization-based approaches (such as EWC, OWM, and L2) and replay-based approaches (such as A-GEM and DER++) perform generally better in BF+CNN than in Adapter-BERT, due to the fewer parameters used to update models and apply regularization on them.

KAN and HAT require problem identifier, and suffer from CF in TIL. We extended them to DIL by using the last model, which, however, shows low results in DIL. Similarly, also CAT (which extends HAT), SRK and UCL cannot achieve competitive results. Approaches specifically designed for CL in NLP (*i.e.*, B-CL and CTR) show clear improvements compared to the others. B-CL and CTR have been mainly designed for TIL: they achieve competitive results in TIL setup, however they fail when employed in DIL. HOP outperforms or it is comparable to the current state-of-the-art competitors in every scenario, and it can deal both with large scale data (*e.g.*, DSC full) and with limited data (*e.g.*, DSC small) in both TIL and DIL. We confirm these findings by looking at the aggregate results reported in Table 2. The results show that HOP robustly outperforms all

	CL Method	**Avg Benchmarks**					
		TIL		**DIL**		**Avg Setups**	
		mAcc	**MF1**	**mAcc**	**MF1**	**mAcc**	**MF1**
BF+Lin	SDL	66.32	57.84	58.40	53.00	62.36	55.42
	FT	60.19	50.13	61.31	56.45	60.75	53.29
	HOP (ours)	**81.39**	**78.89**	**76.56**	**74.47**	**78.98**	**76.68**
Bert (frozen) + CNN	SDL	80.41	74.05	80.41	74.05	80.41	74.05
	FT	77.15	71.03	78.84	75.67	78.00	73.35
	L2	62.79	54.26	72.95	69.54	67.87	61.90
	A-GEM	76.26	70.20	75.70	71.37	75.98	70.79
	DER++	74.84	69.29	76.74	71.86	75.79	70.58
	EWC	78.13	73.09	80.15	77.37	79.14	75.23
	OWM	77.63	72.40	72.93	67.34	75.28	69.87
	UCL	81.01	76.90	80.73	77.68	80.87	77.29
	HAT	83.36	79.94	75.53	71.27	79.44	75.61
	CAT	81.20	75.72	73.25	70.15	77.23	72.94
	KAN	78.18	74.73	75.25	71.53	76.72	73.13
	SRK	79.36	76.98	75.22	71.65	77.29	74.32
	HOP (ours)	**85.80**	**83.67**	**82.82**	**80.97**	**84.31**	**82.32**
Adapter-BERT	SDL	83.86	80.77	83.86	80.77	83.86	80.77
	FT	62.35	54.60	80.48	77.41	71.42	66.00
	L2	70.92	67.08	78.22	74.27	74.57	70.67
	A-GEM	63.12	56.16	79.85	71.42	71.48	63.79
	DER++	60.70	55.74	79.95	77.38	70.32	66.56
	EWC	62.98	56.65	78.90	74.10	70.94	65.37
	OWM	74.79	71.51	77.73	75.16	76.26	73.33
	UCL	59.63	47.41	69.31	59.46	64.47	53.44
	HAT	84.34	81.65	81.06	78.35	82.70	80.00
	B-CL	84.08	81.76	81.92	79.94	83.00	80.85
	CTR	86.79	85.21	82.01	80.05	84.40	82.63
	HOP (ours)	**88.52**	**87.13**	**84.48**	**82.48**	**86.50**	**84.80**

Table 2: Aggregate results from Table 1. First (second) vertical group of 2 columns: results averaged over the 5 benchmarks for TIL (DIL). Last vertical group: averaged over both benchmarks and CL setups.

baseline competitors in both TIL and DIL (first and second vertical groups). Also, CIL-based methods are inadequate for TIL and DIL in NLP. Finally, the last vertical block provides a further comparison aggregated across all benchmarks and CL setups, and is helpful to grasp an overall sense of the results. In general, the best performing frameworks for BF+CNN are HOP, UCL and HAT; while for Adapter-BERT, they are HOP, CTR, B-CL, and HAT.

CF and KT. We report additional metrics to evaluate the intrinsic CF and KT properties of CL models in Table 3 for both TIL and DIL in the DSC small dataset. Most regularization- and replay-based approaches designed for image classification (first group of eight rows) are inadequate to address CL in NLP. These methods show low accuracy due to high forgetting and low KT (BwT and FwT), despite having good plasticity (Pla) to learn representations for a new problem. Methods designed for CL in NLP (second group of three rows), instead, can effectively increase accuracy ($mAcc$ and $MF1$) by increasing KT, reducing $Forg$ whilst maintaining Pla. Compared to competitors, our HOP can find a better balance between CF and KT. In both TIL and DIL, modeling high order statistics using HOP leads to increased $mAcc$ and $MF1$ by reducing $Forg$, although showing comparable or more conservative results in terms of KT properties (BwT, FwT and Pla of HOP are not always maximized). Overall, our framework achieves a better trade-off and outperforms methods proposed specifically for TIL and for DIL.

HOP Improves Other CL Methods. To ensure that HOP is beneficial to CL in NLP applications, we include it in competing CL methods and report the results in Table 4 for DSC small in both TIL and DIL. Comparing the results against Table 1 (gains are reported within brackets in Table 4 for convenience) emerges clear how HOP improves CL methods almost every time, only exception for EWC DIL. In some cases, we observe a large gain up to about 140%. The gain is experienced in both TIL and DIL, the former being more largely improved by our HOP. The integrated methods robustly outperform the original methods along all the evaluated metrics. Remarkably, also current state-of-the-art approaches as B-CL and CTR are significantly improved by our framework.

Per-Problem Acc is shown in Figure 3. Acc evolution over problem is measured by $mAcc_t$ (*i.e.*, per-problem accuracy averaged over all problems) and $mAcc_{t,\leq T}$ (*i.e.*, per-problem accuracy averaged over the problems seen so far). FT exhibits a clear performance drop due to CF and inability to perform KT. Methods designed for CL in NLP show an almost perfect monotonically increasing behaviour of $mAcc_{t,\leq T}$, since they are capable of learning new problems (high plasticity) without forgetting previous ones.

Efficiency. HOP only adds a small increase in parameters and computation time. We observe in Table 3 that adapters of Adapter-BERT account only for about 40% (73.8M) of the total number of parameters (183.3M), while clearly outperforming architectures with only a linear or convolution-based trainable head (see Table 1). Compared to FT, HOP introduces just about 3% more total parameters, increasing the average running training time per problem by about 8% (1.2 to 1.3 min). We confirm in Table 4 that HOP only adds a minimal increase in computation time when added on top of existing CL methods. On average, HOP increases the mean running time per problem by just 7.2%.

We further highlight the effectiveness of HOP in Figure 4. Our approach is more computationally efficient than existing methods while outperforming them in terms of accuracy. In particular, HOP is much faster than the main competitors being about $24\times$ faster than CTR, $4\times$ faster than B-CL.

Other Pooling Schemes and Order of HOP. Next, we observe in Table 5 how popular pooling schemes underperform our solution. As a baseline, `[CLS]` token is used for the final classification (Devlin et al. 2019): its low results are due to the variable distribution of tokens over training. AVG pooling (LeCun et al. 1998) already shows remarkable improvements especially in handling the problem variability in TIL. MAX pooling (Riesenhuber and Poggio 1999) has a slightly worse effect than AVG. Concatenating AVG and MAX (AVGMAX, Monteiro, Alam, and Falk 2020) improves compared to using single clues alone. Employment of second order statistics of tokens alone, *i.e.* either standard

	TIL						DIL								
	mAcc↑	**MF1↑**	**BwT↑**	**FwT↑**	**Forg↓**	**Pla↑**	**mAcc↑**	**MF1↑**	**BwT↑**	**FwT↑**	**Forg↓**	**Pla↑**	**#OP↓**	**#TP↓**	**Time↓**
FT	55.19	35.28	59.59	45.50	28.79	74.71	82.22	81.32	83.17	79.24	5.56	82.99	183.3	73.8	1.2
L2	70.87	69.11	71.48	57.46	13.60	82.86	83.78	82.86	83.48	81.22	1.44	82.99	183.3	73.8	1.4
A-GEM	59.35	54.20	56.77	46.14	26.05	84.24	86.25	85.49	84.92	83.03	0.90	84.80	183.3	73.8	8.7
DER++	63.11	61.96	50.92	40.26	22.76	83.31	84.00	83.16	85.06	80.97	3.46	84.15	183.3	73.8	1.2
EWC	58.34	42.85	54.15	43.10	25.65	79.13	87.29	86.34	86.05	82.25	0.83	83.19	183.3	73.8	1.3
OWM	73.97	71.96	72.19	58.32	12.40	83.39	77.32	76.68	73.78	54.12	6.76	79.87	184.4	74.8	1.4
UCL	48.36	32.07	51.19	49.21	7.48	52.12	56.53	52.96	51.07	43.16	7.08	57.71	183.4	73.9	1.3
HAT	80.83	78.41	75.97	58.15	6.50	83.12	86.22	85.44	85.25	80.57	1.57	84.26	184.0	74.5	1.6
B-CL	84.34	83.12	84.76	48.40	0.67	81.98	85.92	85.11	85.18	78.73	1.34	85.92	220.2	110.7	4.8
CTR	83.96	83.00	84.01	47.95	0.28	82.86	86.01	85.16	85.26	78.81	1.56	85.89	186.6	77.1	30.8
HOP (ours)	85.63	84.18	84.39	45.32	0.57	82.76	**87.84**	**86.96**	85.11	82.33	0.39	84.41	189.2	79.7	1.3
B-CL + HOP	86.93	86.25	86.84	51.47	0.24	86.53	87.08	86.30	86.47	84.49	0.88	87.22	244.8	134.3	4.9
CTR + HOP	**87.08**	**86.32**	86.67	57.04	0.49	86.33	86.54	85.73	85.73	81.72	1.44	86.26	210.2	100.8	32.7

Table 3: Collection of metrics on the DSC small dataset for every competing approach on both TIL and DIL setups.

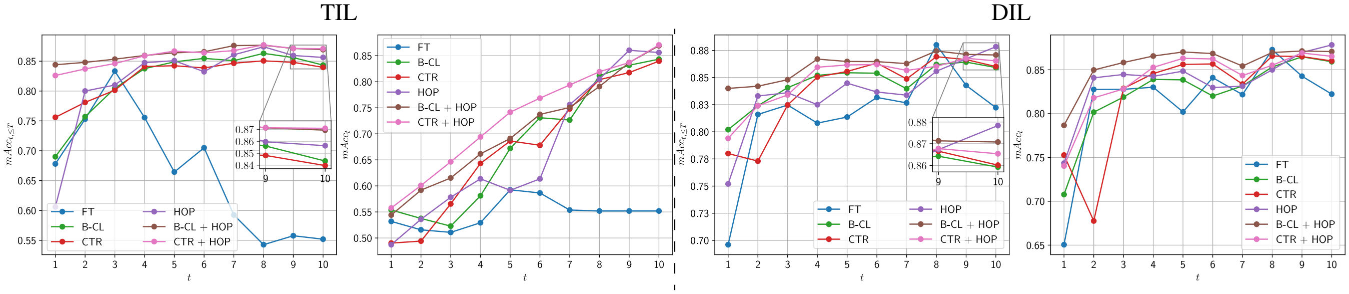


Figure 3: Per-problem accuracy ($mAcc_t$ and $mAcc_{t,\leq T}$) on the DSC small dataset for both TIL and DIL setups.

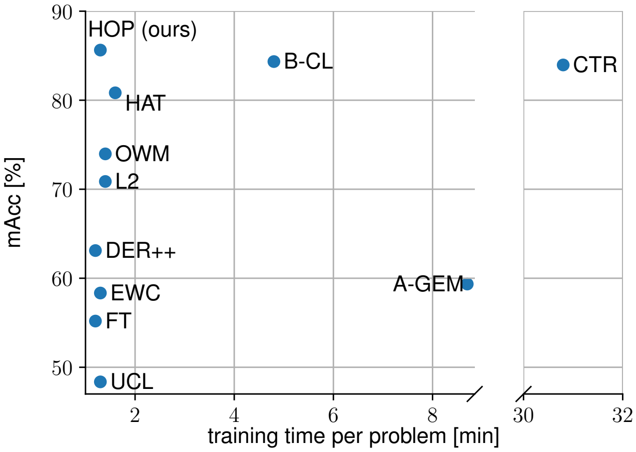


Figure 4: mAcc vs. training time per problem on the TIL setup. Optimal results are in the top-left corner.

deviation (TSDP, Wang et al. 2021) or covariance (iSQRT-COV, Li et al. 2018), improves on TIL but not on DIL compared to the baseline. HOP with $p = 2$ improves results compared to AVGMAX whilst using the same number of statistical measures from the distribution of tokens. We observe that the best results are obtained for HOP with $p = 3$. Intuitively, features of different problems have similar distribution of first moments, while higher moments are discriminative for the specific problems. We compute the average Wasserstein distance between distributions of features of different problems to quantify this effect. The mean distance of first moments (0.15 ± 0.03) is considerably lower than the mean distance of second (0.36 ± 0.05) and third moments (0.26 ± 0.07), indicating that problems are more entangled in the feature space of first moment (lower distance) than of second-third moments. Thus, second-third moments improve accuracy. On the other hand, moments > 3 show lower distance (*e.g.*, the average distance of fourth moments is 0.04 ± 0.01) and yield lower results (Table 5). Therefore, they have been ignored. HOP with $m_1 =$ [CLS] concatenates the [CLS] token with high order statistics and shows results similar to our framework, suggesting that the [CLS] token can be used in conjunction with high order statistics in par with AVG. In other words, in HOP, m_1 can be either AVG or [CLS].

Conclusion

We proposed HOP, which, to our knowledge, is the first CL method for both TIL and DIL in various NLP applications (ASC, DSC, NLI, TC). HOP is a novel approach to adapt a pre-trained NLP model for CL. HOP relies on adapter modules and auxiliary MLPs specialized for each problem. Then, it computes high order moments of embedded tokens to extract rich sentence-wide information, op-

HOP +	**TIL**		**DIL**		
	mAcc	**MF1**	**mAcc**	**MF1**	**Time [min]**
FT	85.63 (+55.2%)	84.18 (+138.6%)	**87.94** (+7.0%)	**87.04** (+7.0%)	1.3 (+8.3%)
L2	82.57 (+16.5%)	80.02 (+15.8%)	85.60 (+2.2%)	84.84 (+2.4%)	1.4 (+0.0%)
A-GEM	86.87 (+46.4%)	85.96 (+58.6%)	86.91 (+0.8 %)	86.08 (+0.7%)	8.8 (+1.1%)
DER++	85.49 (+35.5%)	83.97 (+35.5%)	87.00 (+3.6%)	86.05 (+3.5%)	1.5 (+25.0%)
EWC	84.43 (+44.7%)	83.30 (+94.4%)	86.91 (-0.4%)	86.14 (-0.2 %)	1.4 (+7.7%)
B-CL	86.93 (+3.1%)	86.25 (+3.8%)	87.08 (+1.4%)	86.30 (+1.4%)	4.9 (+2.1%)
CTR	**87.08** (+3.7%)	**86.32** (+4.0%)	86.54 (+0.6%)	85.73 (+0.7%)	32.7 (+6.2%)

Table 4: Combination of HOP with competing CL methods on the DSC small dataset.

	TIL		**DIL**	
	mAcc	**MF1**	**mAcc**	**MF1**
[CLS]	55.19	35.28	82.22	81.32
AVG	81.44	80.53	82.94	81.68
MAX	79.49	78.48	82.63	81.26
AVGMAX	81.51	80.47	83.34	82.27
TSDP	77.52	76.30	78.91	77.50
iSQRT-COV	81.47	80.59	79.62	78.06
HOP $p = 2$ (ours)	83.52	82.53	86.91	86.12
HOP $p = 3$ (ours)	**85.63**	84.18	**87.84**	**86.96**
HOP $p = 4$ (ours)	84.47	83.58	86.65	85.81
HOP with $m_1 =$ [CLS] (ours)	85.47	**84.61**	87.50	86.70

Table 5: Ablation results on DSC small for different pooling schemes and HOP with different values of R.

posed to relying on a single token for classification (*e.g.*, [CLS]), which fails to adapt to dynamic non-stationary input distributions. HOP encourages KT among problems and protects problem-specific knowledge reducing CF. Experiments show that HOP sets new state-of-the-art results on the most widely used CL NLP scenarios. At the same time, HOP only adds a minimal computation footprint.

References

Ács, J.; Kádár, A.; and Kornai, A. 2021. Subword Pooling Makes a Difference. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Association for Computational Linguistics.

Ahn, H.; Cha, S.; Lee, D.; and Moon, T. 2019. Uncertainty-based continual learning with adaptive regularization. *Advances in Neural Information Processing Systems (NeurIPS)*, 32.

Asghar, N.; Mou, L.; Selby, K. A.; Pantasdo, K. D.; Poupart, P.; and Jiang, X. 2020. Progressive Memory Banks for Incremental Domain Adaptation. In *International Conference on Learning Representations (ICLR)*.

Biesialska, M.; Biesialska, K.; and Costa-jussà, M. R. 2020. Continual Lifelong Learning in Natural Language Processing: A Survey. In *International Conference on Computational Linguistics (COLING)*, 6523–6541.

Buzzega, P.; Boschini, M.; Porrello, A.; Abati, D.; and Calderara, S. 2020. Dark experience for general continual learning: a strong, simple baseline. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 15920–15930.

Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka, E. R.; and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In *AAAI Conference on Artificial Intelligence*.

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2018. Efficient Lifelong Learning with A-GEM. In *International Conference on Learning Representations (ICLR)*.

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

Chen, Z.; Ma, N.; and Liu, B. 2015. Lifelong Learning for Sentiment Classification. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, 750–756.

Chien, J.-T.; and Chen, Y.-H. 2021. Continuous-time attention for sequential learning. *AAAI Conference on Artificial Intelligence*, 35(8): 7116–7124.

Chuang, Y.-S.; Su, S.-Y.; and Chen, Y.-N. 2020. Lifelong Language Knowledge Distillation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Cortes, C.; and Mohri, M. 2004. Distribution kernels based on moments of counts. In *International Conference on Machine Learning (ICML)*, 25.

De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; and Tuytelaars, T. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(7): 3366–3385.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4171–4186.

Ding, X.; Liu, B.; and Yu, P. S. 2008. A holistic lexicon-based approach to opinion mining. In *International Conference on Web Search and Data Mining*, 231–240.

Greco, C.; Plank, B.; Fernández, R.; and Bernardi, R. 2019. Psycholinguistics Meets Continual Learning: Measuring Catastrophic Forgetting in Visual Question Answering. In *Annual Meeting of the Association for Computational Linguistics*, 3601–3605.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *Proceeding of the International Conference on Machine Learning (ICML)*, 2790–2799. PMLR.

Hu, M.; and Liu, B. 2004. Mining and summarizing customer reviews. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 168–177.

Hu, W.; Qin, Q.; Wang, M.; Ma, J.; and Liu, B. 2021. Continual learning by using information of each class holistically. *AAAI Conference on Artificial Intelligence*, 35(9).

Isele, D.; and Cosgun, A. 2018. Selective experience replay for lifelong learning. In *AAAI Conference on Artificial Intelligence*, volume 32.

Jung, H.; Ju, J.; Jung, M.; and Kim, J. 2016. Less-forgetting learning in deep neural networks. *arXiv:1607.00122*.

Ke, Z.; and Liu, B. 2022. Continual Learning of Natural Language Processing Tasks: A Survey. *arXiv:2211.12701*.

Ke, Z.; Liu, B.; and Huang, X. 2020. Continual learning of a mixed sequence of similar and dissimilar tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33.

Ke, Z.; Liu, B.; Ma, N.; Xu, H.; and Shu, L. 2021a. Achieving Forgetting Prevention and Knowledge Transfer in Continual Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 22443–22456.

Ke, Z.; Liu, B.; Wang, H.; and Shu, L. 2020. Continual learning with knowledge transfer for sentiment classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 683–698. Springer.

Ke, Z.; Liu, B.; Xu, H.; and Shu, L. 2021b. CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6871–6883.

Ke, Z.; Xu, H.; and Liu, B. 2021. Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4746–4755.

Khayrallah, H.; Thompson, B.; Duh, K.; and Koehn, P. 2018. Regularized Training Objective for Continued Training for Domain Adaptation in Neural Machine Translation. In *Workshop on Neural Machine Translation and Generation*, 36–44. Melbourne, Australia: Association for Computational Linguistics.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. Doha, Qatar: Association for Computational Linguistics.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13): 3521–3526.

Lang, K. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings*, 331–339. Elsevier.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11): 2278–2324.

Lesort, T.; Lomonaco, V.; Stoian, A.; Maltoni, D.; Filliat, D.; and Díaz-Rodríguez, N. 2020. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58: 52–68.

Li, P.; Xie, J.; Wang, Q.; and Gao, Z. 2018. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 947–955.

Li, Y.; Zhao, L.; Church, K.; and Elhoseiny, M. 2019. Compositional Language Continual Learning. In *International Conference on Learning Representations (ICLR)*.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12): 2935–2947.

Liu, Q.; Gao, Z.; Liu, B.; and Zhang, Y. 2015. Automated rule selection for aspect extraction in opinion mining. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Liu, T.; Ungar, L.; and Sedoc, J. 2019. Continual Learning for Sentence Representations Using Conceptors. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3274–3279.

Liu, Z.; Winata, G. I.; Madotto, A.; and Fung, P. 2020. Exploring fine-tuning techniques for pre-trained cross-lingual models via continual learning. *arXiv:2004.14218*.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

Lv, G.; Wang, S.; Liu, B.; Chen, E.; and Zhang, K. 2019. Sentiment classification by leveraging the shared knowledge from a sequence of domains. In *International Conference on Database Systems for Advanced Applications*, 795–811. Springer.

Madotto, A.; Lin, Z.; Zhou, Z.; Moon, S.; Crook, P.; Liu, B.; Yu, Z.; Cho, E.; and Wang, Z. 2020. Continual learning in task-oriented dialogue systems. *arXiv:2012.15504*.

Mai, Z.; Li, R.; Jeong, J.; Quispe, D.; Kim, H.; and Sanner, S. 2022. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469: 28–51.

Mallya, A.; Davis, D.; and Lazebnik, S. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *European Conference on Computer Vision (ECCV)*, 67–82.

Mallya, A.; and Lazebnik, S. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 7765–7773.

Maracani, A.; Michieli, U.; Toldo, M.; and Zanuttigh, P. 2021. Recall: Replay-based continual learning in semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 7026–7035.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, 109–165. Elsevier.

Michieli, U.; Moon, J.; Kim, D.; and Ozay, M. 2024. Object-Conditioned Bag of Instances for Few-Shot Personalized Instance Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Michieli, U.; and Ozay, M. 2023. Online Continual Learning for Robust Indoor Object Recognition. *International Conference on Intelligent Robotics*.

Michieli, U.; Parada, P. P.; and Ozay, M. 2023. Online Continual Learning in Keyword Spotting for Low-Resource Devices via Pooling High-Order Temporal Statistics. *INTERSPEECH*.

Michieli, U.; Toldo, M.; and Zanuttigh, P. 2022. Domain adaptation and continual learning in semantic segmentation. In Davies, E.; and Turk, M. A., eds., *Advanced Methods and Deep Learning in Computer Vision*, Computer Vision and Pattern Recognition, 275–303. Academic Press. ISBN 978-0-12-822109-9.

Michieli, U.; and Zanuttigh, P. 2019. Incremental learning techniques for semantic segmentation. In *International Conference on Computer Vision Workshops (ICCVW)*, 0–0.

Michieli, U.; and Zanuttigh, P. 2021. Knowledge distillation for incremental learning in semantic segmentation. *Computer Vision and Image Understanding (CVIU)*, 205: 103167.

Mirzadeh, S. I.; Chaudhry, A.; Yin, D.; Nguyen, T.; Pascanu, R.; Gorur, D.; and Farajtabar, M. 2022. Architecture matters in continual learning. *arXiv:2202.00275*.

Monaikul, N.; Castellucci, G.; Filice, S.; and Rokhlenko, O. 2021. Continual learning for named entity recognition. *AAAI Conference on Artificial Intelligence*, 35(15): 13570–13577.

Monteiro, J.; Alam, M. J.; and Falk, T. 2020. On the performance of time-pooling strategies for end-to-end spoken language identification. In *Conference on Language Resources and Evaluation Conference (LREC)*, 3566–3572.

Nguyen, C. V.; Li, Y.; Bui, T. D.; and Turner, R. E. 2018. Variational Continual Learning. In *International Conference on Learning Representations (ICLR)*.

Papoulis, A.; and U. Pillai, S. 2002. *Probability, random variables and stochastic processes*.

Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; and Gurevych, I. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 487–503. Association for Computational Linguistics (ACL).

Pfeiffer, J.; Vulić, I.; Gurevych, I.; and Ruder, S. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7654–7673.

Pontiki, M.; Galanis, D.; Pavlopoulos, J.; Papageorgiou, H.; Androutsopoulos, I.; and Manandhar, S. 2014. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *PInternational Workshop on Semantic Evaluation (SemEval 2014)*, 27–35. Dublin, Ireland: Association for Computational Linguistics.

Qian, K.; Wei, W.; and Yu, Z. 2021. A student-teacher architecture for dialog domain adaptation under the meta-learning setting. *AAAI Conference on Artificial Intelligence*, 35(15): 13692–13700.

Qin, Q.; Hu, W.; and Liu, B. 2020. Using the past knowledge to improve sentiment classification. In *Findings of the Association for Computational Linguistics: Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1124–1133.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupeke2022continualpretrainingrvised multitask learners. *OpenAI blog*, 1(8): 9.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001–2010.

Riesenhuber, M.; and Poggio, T. 1999. Hierarchical models of object recognition in cortex. *Nature Neural Networks (NN)*, 2(11): 1019–1025.

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

Ruvolo, P.; and Eaton, E. 2013. ELLA: An efficient lifelong learning algorithm. In *International Conference on Machine Learning (ICML)*, 507–515. PMLR.

Serra, J.; Suris, D.; Miron, M.; and Karatzoglou, A. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, 4548–4557. PMLR.

Shen, Y.; Zeng, X.; and Jin, H. 2019. A progressive model to enable continual learning for semantic slot filling. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1279–1284.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

Silver, D. L.; Yang, Q.; and Li, L. 2013. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI spring symposium series*.

Sun, F.-K.; Ho, C.-H.; and Lee, H.-Y. 2019. LAMOL: LAnguage MOdeling for Lifelong Language Learning. In *International Conference on Learning Representations (ICLR)*.

Van de Ven, G. M.; and Tolias, A. S. 2019. Three scenarios for continual learning. *arXiv:1904.07734*.

Wang, H.; Liu, B.; Wang, S.; Ma, N.; and Yang, Y. 2019. Forward and backward knowledge transfer for sentiment classification. In *Asian Conference on Machine Learning (ACML)*, 457–472. PMLR.

Wang, J.; Dong, D.; Shou, L.; Chen, K.; and Chen, G. 2023a. Effective Continual Learning for Text Classification with Lightweight Snapshots. *AAAI Conference on Artificial Intelligence*, 37(8): 10122–10130.

Wang, S.; Lv, G.; Mazumder, S.; Fei, G.; and Liu, B. 2018. Lifelong learning memory networks for aspect sentiment classification. In *IEEE International Conference on Big Data (Big Data)*, 861–870. IEEE.

Wang, S.; Yang, Y.; Qian, Y.; and Yu, K. 2021. Revisiting the statistics pooling layer in deep speaker embedding learning. In *International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 1–5. IEEE.

Wang, Y.; Ma, Z.; Huang, Z.; Wang, Y.; Su, Z.; and Hong, X. 2023b. Isolation and impartial aggregation: A paradigm of incremental learning without interference. *AAAI Conference on Artificial Intelligence*, 37(8): 10209–10217.

Williams, A.; Nangia, N.; and Bowman, S. R. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Wu, C.; Wu, F.; Qi, T.; Cui, X.; and Huang, Y. 2020. Attentive pooling with learnable norms for text representation. In *Annual Meeting of the Association for Computational Linguistics*, 2961–2970.

Wu, T.; Liu, Q.; Cao, Y.; Huang, Y.; Wu, X.-M.; and Ding, J. 2023. Continual Graph Convolutional Network for Text Classification. *AAAI Conference on Artificial Intelligence*.

Xu, J.; and Zhu, Z. 2018. Reinforced continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 31.

Zeng, G.; Chen, Y.; Cui, B.; and Yu, S. 2019. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8): 364–372.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, 3987–3995. PMLR.

Zhan, R.; Liu, X.; Wong, D. F.; and Chao, L. S. 2021. Meta-curriculum learning for domain adaptation in neural machine translation. *AAAI Conference on Artificial Intelligence*, 35(16): 14310–14318.

Zhao, S.; You, F.; Chang, W.; Zhang, T.; and Hu, M. 2022. Augment BERT with average pooling layer for Chinese summary generation. *Journal of Intelligent & Fuzzy Systems*, 1–10.

Zhou, Y.; Zhu, F.; Song, P.; Han, J.; Guo, T.; and Hu, S. 2021. An adaptive hybrid framework for cross-domain aspect-based sentiment analysis. *AAAI Conference on Artificial Intelligence*, 35(16): 14630–14637.