Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks

Zixuan Ke[1], **Hu Xu**[2] and **Bing Liu**[1]
[1]Department of Computer Science, University of Illinois at Chicago
[2]Facebook AI Research
[1]{zke4,liub}@uic.edu
[2]huxu@fb.com

Abstract

This paper studies continual learning (CL) of a sequence of aspect sentiment classification (ASC) tasks. Although some CL techniques have been proposed for document sentiment classification, we are not aware of any CL work on ASC. A CL system that incrementally learns a sequence of ASC tasks should address the following two issues: (1) transfer knowledge learned from previous tasks to the new task to help it learn a better model, and (2) maintain the performance of the models for previous tasks so that they are not forgotten. This paper proposes a novel capsule network based model called B-CL to address these issues. B-CL markedly improves the ASC performance on both the new task and the old tasks via forward and backward knowledge transfer. The effectiveness of B-CL is demonstrated through extensive experiments.[1]

1 Introduction

Continual learning (CL) aims to incrementally learn a sequence of tasks. Once a task is learned, its training data is often discarded (Chen and Liu, 2018). This is in contrast to *multi-task learning*, which assumes the training data of all tasks are available simultaneously. The CL setting is important in many practical scenarios. For example, a sentiment analysis company typically has many clients and each client often wants to have their private data deleted after use. In the personal assistant or chatbot context, the user does not want his/her chat data, which often contains sentiments or emotions, uploaded to a central server. In such applications, if we want to improve sentiment analysis accuracy for each user/client without breaching confidentiality, CL is a suitable solution.

There are two main types of continual learning: (1) *Task Incremental Learning* (TIL) and (2) *Class Incremental Learning* (CIL). This work focuses on TIL, where each task is a separate *aspect sentiment classification* (ASC) task. An ASC task is defined as follows (Liu, 2015): given an aspect (e.g., *picture quality* in a camera review) and a sentence containing the aspect in a particular domain (e.g., camera), classify if the sentence expresses a positive, negative, or neutral (no opinion) about the aspect. TIL builds a model for each task and all models are in one neural network. In testing, the system knows which task each test instance belongs to and uses only the model for the task to classify the instance. In CIL, each task contains one or more classes to be learned. Only one model is built for all classes. In testing, a test case from any class may be presented to the model to classify without giving it any task information. This setting is not applicable to ASC.

Task ID	Domain/Task	One Training Example(in that domain/task)
1	Vacuum Cleaner [CF]	This vacuum cleaner *sucks* !!!
2	Desktop [KT]	The keyboard is clicky .
3	Tablet [KT]	The soft keyboard is hard to use.
4 (new task)	Laptop	The new keyboard *sucks* and is hard to click!

Table 1: Tasks 2 and 3 have shareable knowledge to transfer (KT) to the new task, whereas Task 1 has specific knowledge that is expected to be isolated from the new task to avoid catastrophic forgetting (CF) (although they use the same word). Note that here we use only one sentence to represent a task, but each task actually represents a domain with all its sentences.

Our goal of this paper is to achieve the following two objectives: (1) transfer the knowledge learned from previous tasks to the new task to help learn a better model for the new task without accessing the training data from previous tasks (in contrast to multi-task learning), and (2) maintain (or even improve) the performance of the old models for previous tasks so that they are not forgotten. The focus of the existing CL (TIL or CIL) research has been on solving (2), *catastrophic forgetting* (CF) (Chen and Liu, 2018; Ke et al., 2020a). CF means that when a network learns a sequence of tasks, the learning of each new task is likely to change the net-

[1]https://github.com/ZixuanKe/PyContinual

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4746–4755
June 6–11, 2021. ©2021 Association for Computational Linguistics

work parameters learned for previous tasks, which degrades the model performance for the previous tasks (McCloskey and Cohen, 1989). In our case, (1) is also important as ASC tasks are similar, i.e., words and phrases used to express sentiments for different products/tasks are similar. To achieve the objectives, the system needs to identify the *shared knowledge* that can be transferred to the new task to help it learn better and the *task specific knowledge* that needs to be protected to avoid forgetting of previous models. Table 1 gives an example.

Fine-tuned BERT (Devlin et al., 2019) is one of the most effective methods for ASC (Xu et al., 2019; Sun et al., 2019). However, our experiments show that it works very poorly for TIL. The main reason is that the fine-tuned BERT on a task/domain captures highly task specific information which is difficult to transfer to a new task.

In this paper, we propose a novel model called B-CL (*BERT-based Continual Learning*) for ASC continual learning. The key novelty is a building block, called *Continual Learning Adapter* (CLA) inspired by the Adapter-BERT in (Houlsby et al., 2019). CLA leverages capsules and dynamic routing (Sabour et al., 2017) to identify previous tasks that are similar to the new task and exploit their shared knowledge to help the new task learning and uses task masks to protect task-specific knowledge to avoid forgetting (CF). We conduct extensive experiments over a wide range of baselines to demonstrate the effectiveness of B-CL.

In summary, this paper makes two key contributions. **(1)** It proposes the problem of task incremental learning for ASC. **(2)** It proposes a new model B-CL with a novel adapter CLA incorporated in a pre-trained BERT to enable ASC continual learning. CLA employs capsules and dynamic routing to explore and transfer relevant knowledge from old tasks to the new task and uses task masks to isolate task-specific knowledge to avoid CF. To our knowledge, none of these has been done before.

2 Related Work

Continual learning (CL) has been studied extensively (Chen and Liu, 2018; Parisi et al., 2019). To our knowledge, no existing work has been done on CL for a sequence of ASC tasks, although CL of a sequence of document sentiment classification tasks has been done.

Continual Learning. Existing work has mainly focused on dealing with catastrophic forgetting (CF).

Regularization-based methods, such as those in (Kirkpatrick et al., 2016; Lee et al.; Seff et al., 2017), add a regularization in the loss to consolidate previous knowledge when learning a new task.

Parameter isolation-based methods, such as those in (Serrà et al., 2018; Mallya and Lazebnik, 2018; Fernando et al., 2017), make different subsets of the model parameters dedicated to different tasks and identify and mask them out during the training of the new task.

Gradient projection-based method, such as that in (Zeng et al., 2019), ensures the gradient updates occur only in the orthogonal direction to the input of the old tasks and thus will not affect old tasks.

Replay-based methods, such as those in (Rebuffi et al., 2017; Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2019), retain an exemplar set of old task training data to help train the new task. The methods in (Shin et al., 2017; Kamra et al., 2017; Rostami et al., 2019; He and Jaeger, 2018) build data generators for previous tasks so that in learning the new task, they can use some generated data for previous tasks to help avoid forgetting.

As these methods are mainly for avoiding CF, after learning a sequence of tasks, their final models are typically worse than learning each task separately. The proposed B-CL not only deals with CF, but also performs knowledge transfer to improve the performance of both the new and the old tasks.

Lifelong Learning (LL). LL is now regarded the same as CL, but early LL mainly aimed at improving the new task learning through forward transfer without tackling CF (Silver et al., 2013; Ruvolo and Eaton, 2013; Chen and Liu, 2018).

Several researchers have used LL for document-level sentiment classification. Chen et al. (2015) and Wang et al. (2019) proposed two Naive Bayes (NB) approaches to help improve the new task learning. A heuristic NB method was also used in (Wang et al., 2019). Xia et al. (2017) presented a LL approach based on voting of individual task classifiers. All these works do not use neural networks, and are not concerned with the CF problem.

Shu et al. (2017) used LL for aspect extraction, which is a different problem. Wang et al. (2018) used LL for ASC, but improved only the new task and did not deal with CF. Existing CL systems SRK (Lv et al., 2019), KAN (Ke et al., 2020b) and L2PG (Qin et al., 2020) are for document sentiment classification, but not ASC. Ke et al. (2020a) also performed transfer in the image domain.

Recently, capsule networks (Hinton et al., 2011) have been used in sentiment classification and text classification (Chen and Qian, 2019; Zhao et al., 2019). But they have not been used in CL.

3 Preliminary

This section introduces BERT, Adapter-BERT and Capsule Network as they are used in our model.

BERT for ASC. Due to its superior performance, this work uses BERT (Devlin et al., 2019) and its transformer (Vaswani et al., 2017) architecture as the base. We also adopt the ASC formulation in (Xu et al., 2019), where the aspect term and review sentence are concatenated via `[SEP]`. The sentiment polarity is predicted on top of the `[CLS]` token. Although BERT can achieve impressive performance on a single ASC task, its architecture and fine-tuning paradigm are not suitable for CL (see Sec. 1). Experiments show that it performs very poorly for CL (Sec. 5.4). We found that Adapter-BERT (Houlsby et al., 2019) is a better fit for CL.

Adapter-BERT. Adapter-BERT basically inserts a 2-layer fully-connected network (adapter) in each transformer layer of BERT (see Figure 1(A)). During training for the end-task, only the adapters and normalization layers are trained, no change to any other BERT parameters, which is good for CL because fine-tuning BERT itself causes serious forgetting. Adapter-BERT achieves similar performances to fine-tuned BERT (Houlsby et al., 2019). We propose to exploit the adapter idea and the capsule network to achieve effective CL for ASC tasks.

Capsule Network. Capsule network (CapsNet) is a relatively new classification architecture (Hinton et al., 2011; Sabour et al., 2017). Unlike CNN, CapsNet replaces the scalar feature detectors with vector capsules that can preserve additional information such as position and thickness in images. A typical CapsNet has two capsule layers. The primary layer stores low-level feature maps and the class layer produces the probability for classification with each capsule corresponding to one class. It uses a dynamic routing algorithm to enable each lower level capsule to send its output to the similar (or "agreed", computed by dot product) higher level capsule. This is the key property that we exploit to identify and group similar tasks and their shared features or knowledge.

Note that the proposed B-CL does not adopt the whole capsule network as we are only interested in the capsule layers and dynamic routing instead of the max-margin loss and the classifier.

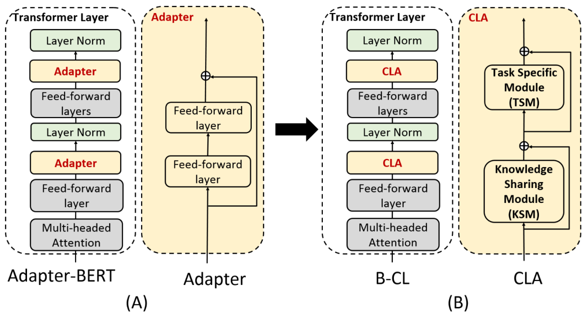

Figure 1: **(A).** Adapter-BERT (Houlsby et al., 2019) and its adapters in a transformer (Vaswani et al., 2017) layer. An adapter is a 2-layer fully connected network with a skip-connection. It is added twice to each Transformer layer. Only the adapters (yellow boxes) and layer norm (green boxes) layers are trainable. The other modules (grey boxes) are frozen. **(B).** Proposed B-CL, which replaces the adapter with CLA. CLA has two sub-modules: knowledge sharing module (KSM) and task specific module (TSM). Each of these modules has a skip-connection.

4 Continual Learning Adapter (CLA)

Recall the proposed B-CL aims to achieve (1) knowledge transfer between related old tasks and the new task through knowledge sharing and (2) forgetting avoidance through preventing task specific knowledge of previous tasks from being overwritten by the new task learning. Inspired by Adapter-BERT, we propose the *continual learning adapters* (CLA) to replace the adapters in Adapter-BERT to enable CL as in Figure 1(B) to achieve BERT based continual learning for ASC.

The architecture of CLA is shown in Figure 2(A). It contains two modules: (1) *knowledge sharing module* (KSM) for identifying and exploiting shareable knowledge from the similar previous tasks and the new task, and (2) *task specific module* (TSM) for learning task specific neurons and protecting them from being updated by the new task.

CLA takes two inputs: (1) hidden states $h^{(t)}$ from the feed-forward layer inside a transformer layer and (2) task ID t. The outputs are hidden states with features good for the t-th task. KSM leverages capsule layers (see below) and dynamic routing to group similar tasks and the shareable knowledge, whereas TSM takes advantage of task mask (TM) to protect neurons for a particular task and leave other neurons free. Those free neurons are later used by TSM for a new task. Since TMs are differentiable, the whole system B-CL can be trained end-to-end. We detail each module below.

4.1 Knowledge Sharing Module (KSM)

KSM groups similar tasks and shared knowledge (features) among them to enable knowledge transfer among similar tasks. This is achieved through two capsule layers (*task capsule layer* and *knowledge sharing capsule layer*) and the dynamic routing algorithm of the capsule network.

4.1.1 Task Capsule Layer (TCL)

Each capsule in TCL represents a task and TCL prepares low-level features derived from each task (Figure 2(A)). As such, a capsule is added to TCL for every new task. This incremental growing is efficient and easy because these capsules are discrete and do not share parameters. Also each capsule is simply a 2-layer fully connected network with a small number of parameters. Let $h^{(t)} \in \mathbb{R}^{d_t \times d_e}$ be the input of CLA, where d_t is the number of tokens and d_e the number of dimensions. Let the set of tasks learned so far be $\mathcal{T}_{\text{prev}}$ (before learning the new task t) and $|\mathcal{T}_{prev}| = n$. In TCL, we have $n+1$ different capsules representing all past n learned tasks as well as the new task t. The capsule for the i-th ($i \leq n+1$) task is

$$p_i^{(t)} = f_i(h^{(t)}), \tag{1}$$

where $f_i(\cdot) = \text{MLP}_i(\cdot)$ denotes a 2-layer fully-connected network.

4.1.2 Knowledge Sharing Capsule Layer (KCL)

Each *knowledge sharing capsule* in KCL captures those tasks (i.e., their task capsules $\{p_i^{(t)}\}_1^{n+1}$) with similar features or shared knowledge. This is automatically achieved by the *dynamic routing* algorithm. Recall dynamic routing encourages each lower level capsule (task capsule in our case) to send its output to the similar (or "agreed") higher level capsule (knowledge sharing capsule in our case).

Essentially, the similar task capsules (with many shared features) are "clustered" together by higher coefficients (which determine how much a task capsule can go to the next layer) while dissimilar tasks (with few shared features) are blocked via low coefficients. Such clustering identifies the shared features or knowledge from multiple task capsules as well as helps backward transfer across the similar tasks.

KCL first turns each task capsule $p_i^{(t)}$ into a temporary feature $u_{j|i}^{(t)}$ as:

$$u_{j|i}^{(t)} = W_{ij} p_i^{(t)}, \tag{2}$$

where $W_{ij} \in \mathbb{R}^{d_s \times d_k}$ is the weight matrix, d_s and d_k are the dimensions of task capsule i and knowledge sharing capsule j. The number of knowledge sharing capsules is a hyperparameter detailed in the experiment section. The temporary features are summed up with weights $c_{ij}^{(t)}$ to obtain the initial knowledge sharing capsule $s_j^{(t)}$:

$$s_j^{(t)} = \sum_i c_{ij}^{(t)} u_{j|i}^{(t)}, \tag{3}$$

where $c_{ij}^{(t)}$ is a coupling coefficient summed up to 1 and we detail how to compute it later. Note that the task capsule for each task in Eq. 1 is mapped to the knowledge sharing capsule in Eq. 3 and $c_{ij}^{(t)}$ indicates how much or how informative the representation of the i-th task is to the j-th knowledge sharing capsule. As a result, a knowledge sharing capsule can represent diverse sharable knowledge. For those tasks with a very low $c_{ij}^{(t)}$, their representations are less considered in the j-th knowledge sharing capsule. This makes sure only task capsules for tasks that are salient or similar to the new task are used and the others task capsules are ignored (and thus protected) to learn more general shareable knowledge. Recall that the ASC tasks are similar and thus such learning of task sharing features can be very important.

Note that in backpropagation, the dissimilar tasks with low $c_{ij}^{(t)}$ are updated with a low gradient while the similar tasks with high $c_{ij}^{(t)}$ are updated with a larger gradient. This encourages *backward transfer* across similar tasks.

Dynamic Routing. The coupling coefficient in Eq. 3 is essential for the quality of shareable knowledge. This is computed by a "routing softmax":

$$c_{ij}^{(t)} = \frac{\exp(b_{ij}^{(t)})}{\sum_o \exp(b_{io}^{(t)})}, \tag{4}$$

where each b_{ij} is the log prior probability showing how salient or similar a task capsule i is to a knowledge sharing capsule j. It is initialized to 0 indicating no salient connection between them at the beginning. We apply the dynamic routing algorithm in (Sabour et al., 2017) to update b_{ij}:

$$b_{ij}^{(t)} \leftarrow b_{ij}^{(t)} + a_{ij}^{(t)}, \tag{5}$$

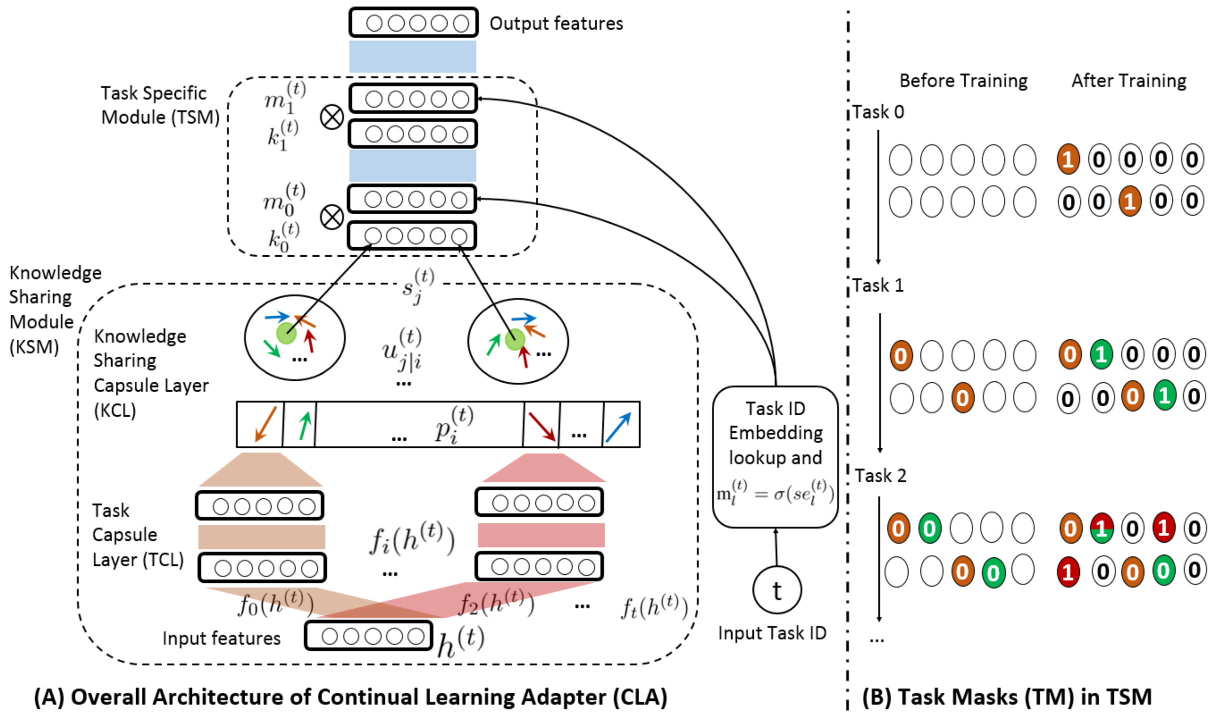

Figure 2: (A) Architecture of CLA: the skip-connection is not shown for clarity. (B) illustration of task masking: a (learnable) task mask is applied after the activation function to *selectively* activate a neuron (or feature). Some notes about (B) are: the two rows of each task corresponds to $k_0^{(t)}$ and $k_1^{(t)}$ in TSM. In the cells before training, those with 0's are the neurons to be protected (masked) and those cells without a number are free neurons (not used). In the cells after training, those cells with 1's show neurons that are important for the current task, which are used as a mask for the future. Those cells with more than one color indicate that they are shared by more than one task. Those 0 cells without a color are not used by any task.

where a_{ij} is the agreement coefficient (see below). Intuitively, this step tends to aggregate the similar (or "agreed") tasks on a knowledge sharing capsule with a higher agreement coefficient a_{ij} and thus a higher logit $b_{ij}^{(t)}$ (Eq. 5) or coupling coefficient $c_{ij}^{(t)}$ (Eq. 4). The agreement coefficient is computed as

$$a_{ij}^{(t)} = u_{j|i}^{(t)} \cdot v_j^{(t)}, \tag{6}$$

where $v_j^{(t)}$ is a normalized representation by applying the non-linear "squash" function (Sabour et al., 2017) to $s_j^{(t)}$ (for the first task, $s_j^{(t)} = u_{j|i}^{(t)}$):

$$v_j^{(t)} = \frac{||s_j^{(t)}||^2}{1 + ||s_j^{(t)}||} \frac{s_j^{(t)}}{||s_j^{(t)}||}, \tag{7}$$

where the length of $v_j^{(t)}$ is normalized to [0,1] to represent the active probability of a knowledge sharing capsule j.

Finally, note that the dynamic routing procedure (Eq. (3)→(7)) is repeated for r iterations.

4.2 Task Specific Module (TSM)

Although knowledge sharing is important for ASC, it is equally important to preserve task specific knowledge for previous tasks to prevent forgetting (CF). To achieve this, we use task masks (Figure 2(B)). Specifically, we first detect the neurons used by each old task, and then block off or mask out all the *used* neurons when learning a new task.

The task specific module consists of differentiable layers (CLA uses a 2-layer fully-connected network). Each layer's output is further applied with a task mask to indicate which neurons should be protected for that task to overcome CF and forbids gradient updates for those neurons during backpropagation for a new task. Those tasks with overlapping masks indicate knowledge sharing. Due to KSM, the features flowing in those overlapping neurons enable the related old tasks to also improve in learning the new task.

4.3 Task Masks

Given the knowledge sharing capsule $s_j^{(t)}$, TSM maps them into input $k_l^{(t)}$ via a fully-connected network, where l is the l-th layer in TSM. A task mask (a "soft" binary mask) $\mathrm{m}_l^{(t)}$ is trained for each task t at each layer l in TSM during training task t's classifier, indicating the neurons that are important for the task in the layer. Here we borrow the hard attention idea in (Serrà et al., 2018) and leverage the task ID embedding to the train the task mask.

For a task ID t, its embedding $e_l^{(t)}$ consists of differentiable deterministic parameters that can be learned together with other parts of the network. It is trained for each layer in TSM. To generate the task mask $\mathrm{m}_l^{(t)}$ from $e_l^{(t)}$, *Sigmoid* is used as a pseudo-gate function and a positive scaling hyper-parameter s is applied to help training. The $m_l^{(t)}$ is computed as follows:

$$m_l^{(t)} = \sigma(se_l^{(t)}). \tag{8}$$

Note that the neurons in $m_l^{(t)}$ may overlap with those in other $m_l^{(i_{\text{prev}})}$s from previous tasks showing some shared knowledge. Given the output of each layer in TSM, $k_l^{(t)}$, we element-wise multiply $k_l^{(t)} \otimes m_l^{(t)}$. The masked output of the last layer $k^{(t)}$ is fed to the next layer of the BERT with a skip-connection (see Figure 1). After learning task t, the final $m_l^{(t)}$ is saved and added to the set $\{m_l^{(t)}\}$.

4.4 Training

For each past task $i_{\text{prev}} \in \mathcal{T}_{\text{prev}}$, its mask $m_l^{(i_{\text{prev}})}$ indicates which neurons are used by that task and need to be protected. In learning task t, $m_l^{(i_{\text{prev}})}$ is used to set the gradient $g_l^{(t)}$ on *all* used neurons of the layer l in TSM to 0. Before modifying the gradient, we first accumulate all used neurons by all previous tasks' masks. Since $m_l^{(i_{\text{prev}})}$ is binary, we use max-pooling to achieve the accumulation:

$$m_l^{(t_{\text{ac}})} = \text{MaxPool}(\{m_l^{(i_{\text{prev}})}\}). \tag{9}$$

The term $m_l^{(t_{\text{ac}})}$ is applied to the gradient:

$$g_l^{'(t)} = g_l^{(t)} \otimes (1 - m_l^{(t_{\text{ac}})}). \tag{10}$$

Those gradients corresponding to the 1 entries in $m_l^{(t_{\text{ac}})}$ are set to 0 while the others remain unchanged. In this way, neurons in an old task are protected. Note that we expand (copy) the vector $m_l^{(t_{\text{ac}})}$ to match the dimensions of $g_l^{(t)}$.

Though the idea is intuitive, $e_l^{(t)}$ is not easy to train. To make the learning of $e_l^{(t)}$ easier and more stable, an annealing strategy is applied (Serrà et al., 2018). That is, s is annealed during training, inducing a gradient flow and set $s = s_{\max}$ during testing. Eq. 8 approximates a unit step function as the mask, with $m_l^{(t)} \to \{0, 1\}$ when $s \to \infty$. A training epoch starts with all neurons being equally active, which are progressively polarized within the epoch. Specifically, s is annealed as follows:

$$s = \frac{1}{s_{\max}} + (s_{\max} - \frac{1}{s_{\max}})\frac{b-1}{B-1}, \tag{11}$$

where b is the batch index and B is the total number of batches in an epoch.

Illustration. In Figure 2(B), after learning the first task (Task 0), we obtain its useful neurons marked in orange with a 1 in each neuron, which serves as a mask in learning future tasks. In learning task 1, those useful neurons for task 0 are masked (with 0 in those orange neurons or cells on the left). The process also learns the useful neurons for task 1 marked in green with 1's. When task 2 arrives, all important neurons for tasks 0 and 1 are masked, i.e., its mask entries are set to 0 (orange and green before training). After training task 2, we see that task 2 and task 1 have a shared neuron that is important to both of them. The shared neuron is marked in both red and green.

5 Experiments

We now evaluate B-CL by comparing it with both *non-continual learning* and *continual learning* baselines. We follow the standard CL evaluation method in (Lange et al., 2019). We first present B-CL a sequence of aspect sentiment classification (ASC) tasks for it to learn. Once a task is learned, its training data is discarded. After all tasks are learned, we test all task models using their respective test data. In training each task, we use its validation set to decide when to stop training.

5.1 Experiment Datasets

Since B-CL works in the CL setting, we employ a set of 19 ASC datasets (reviews of 19 products) to produce sequences of tasks. Each dataset represents a task. The datasets are from 4 sources: (1) **HL5Domains** (Hu and Liu, 2004) with reviews of 5 products; (2) **Liu3Domains** (Liu et al., 2015) with reviews of 3 products; (3) **Ding9Domains** (Ding et al., 2008) with reviews of 9 products; and (4) **SemEval14** with reviews of 2 products - SemEval 2014 Task 4 for laptop and restaurant. For (1), (2) and (3), we split about 10% of the original data as the validation data, another about 10% of the original data as the testing data. For (4), we use 150 examples from the training set for validation. To be consistent with existing research

Data source	Task/domain	Train	Validation	Test
Liu3domain	Speaker	352	44	44
	Router	245	31	31
	Computer	283	35	36
HL5domain	Nokia6610	271	34	34
	Nikon4300	162	20	21
	Creative	677	85	85
	CanonG3	228	29	29
	ApexAD	343	43	43
Ding9domain	CanonD500	118	15	15
	Canon100	175	22	22
	Diaper	191	24	24
	Hitachi	212	26	27
	Ipod	153	19	20
	Linksys	176	22	23
	MicroMP3	484	61	61
	Nokia6600	362	45	46
	Norton	194	24	25
SemEval14	Rest.	3452	150	1120
	Laptop	2163	150	638

Table 2: Number of examples in each task or dataset. More detailed data statistics are given in the *Appendix*.

(Tang et al., 2016), examples belonging to the conflict polarity (both positive and negative sentiments are expressed about an aspect term) are not used. Statistics of the 19 datasets are given in Table 2.

5.2 Compared Baselines

We use 18 baselines, including both *non-continual learning* and *continual learning* methods.

Non-continual Learning (NL) Baselines: NL setting builds a model for each task independently using a separate network. It clearly has no knowledge transfer or forgetting. We have 3 baselines under NL, **(1) BERT**, **(2) Adapter-BERT** and **(3) W2V** (word2vec embeddings). For **BERT**, we use trainable BERT to perform ASC (see Sec. 3); **Adapter-BERT** adapts the BERT as in (Houlsby et al., 2019), where only the adapter blocks are trainable; **W2V** uses embeddings trained on the Amazon review data in (Xu et al., 2018) using FastText (Grave et al., 2018). We adopt the ASC classification network in (Xue and Li, 2018), which takes both aspect term and review sentence as input.

Continual Learning (CL) Baselines. CL setting includes 3 baselines *without dealing with forgetting* (**WDF**) and 12 baselines from 6 state-of-the art *task incremental learning* (TIL) methods dealing with forgetting. WDF baselines greedily learn a sequence of tasks incrementally without explicitly tackling forgetting or knowledge transfer. The 3 baselines under WDF are also **(4) BERT**, **(5) Adapter-BERT** and **(6) W2V**.

The 6 state-of-the-art CL systems are: KAN, SRK, HAT, UCL, EWC and OWM. **KAN** (Ke et al., 2020b) and **SRK** (Lv et al., 2019) are TIL methods for document sentiment classification. HAT, UCL, EWC and OWM were originally designed for image classification. We replace their original MLP or CNN image classification network with CNN for text classification (Kim, 2014). **HAT** (Serrà et al., 2018) is one of the best TIL methods with almost no forgetting. **UCL** (Ahn et al., 2019) is a latest TIL method. **EWC** (Kirkpatrick et al., 2016) is a popular regularization-based class incremental learning (CIL) method, which was adapted for TIL by only training on the corresponding head of the specific task ID during training and only considering the corresponding head's prediction during testing. **OWM** (Zeng et al., 2019) is a state-of-the-art CIL method, which we also adapt to TIL.

From the 6 systems, we created **6 baselines** using **W2V** embeddings with the aspect term added before the sentence so that the CL methods can take both aspect and the review sentence, and **6 baselines** using **BERT (Frozen)** (which replaces W2V embeddings). Following the BERT formulation in Sec. 3, it can naturally take both aspect and review sentence. Adapter-BERT is not applicable to them as their architecture cannot use an adapter.

5.3 Hyperparameters

Unless otherwise stated, for the task sharing module, we employ 2 layers of fully connected network with dimensions 768 in TCL. We also employ 3 knowledge sharing capsules. The dynamic routing is repeated for 3 iterations. For the task-specific module, We employ the embedding with 2000 dimensions as the final and hidden layer of the TSM. The task ID embeddings have 2000 dimensions. A fully connected layer with softmax output is used as the classification heads in the last layer of the BERT, together with the categorical cross-entropy loss. We use 140 for $s_{\max}$ in Eq. 11, dropout of 0.5 between fully connected layers. The training of BERT, Adapter-BERT and B-CL follow that of (Xu et al., 2019). We adopt $\text{BERT}_{\textbf{BASE}}$ (uncased). The maximum length of the sum of sentence and aspect is set to 128. We use Adam optimizer and set the learning rate to 3e-5. For the SemEval datasets, 10 epochs are used and for all other datasets, 30 epochs are used based on results from validation data. All runs use the batch size 32. For the CL baselines, we train all models with the learning rate of 0.05. We early-stop training when there is no improvement in the validation loss for 5 epochs. The

Scenario	Category	Model	Acc.	MF1
Non-continual Learning	BERT	NL	0.8584	0.7635
	Adapter-BERT	NL	0.8596	0.7807
	W2V	NL	0.7701	0.5189
Continual Learning	BERT	WDF	0.4960	0.4308
	Adapter-BERT	WDF	0.5403	0.4481
	W2V	WDF	0.8269	0.7356
	BERT (Frozen)	KAN	0.8549	0.7738
		SRK	0.8476	0.7852
		EWC	0.8637	0.7452
		UCL	0.8389	0.7482
		OWM	0.8702	0.7931
		HAT	0.8674	0.7816
	W2V	KAN	0.7206	0.4001
		SRK	0.7101	0.3963
		EWC	0.8416	0.7229
		UCL	0.8441	0.7599
		OWM	0.8270	0.7118
		HAT	0.8083	0.6363
	B-CL (forward)		**0.8809**	**0.7993**
	B-CL		**0.8829**	**0.8140**

Table 3: Accuracy (Acc.) and Macro-F1 (MF1) averaged over 5 random sequences of 19 tasks.

batch size is set to 64. For all the CL baselines, we use the code provided by their authors and adopt their original parameters (for EWC, we adopt its TIL variant implemented by (Serrà et al., 2018)).

5.4 Results and Analysis

Since the order of the 19 tasks may have an impact on the final results, we randomly choose and run 5 task sequences and average their results. We compute both accuracy and Macro-F1 over 3 classes of polarities, where Macro-F1 is the major metric as the imbalanced classes introduce biases on accuracy. Table 3 gives the average results of 19 tasks (or datasets) over the 5 random task sequences.

Overall Performance. Table 3 shows that B-CL outperforms all baselines markedly. We discuss the detailed observations below:

(1) For non-continual learning (NL) baselines, BERT and Adapter-BERT perform similarly. W2V is poorer, which is understandable.

(2) Comparing NL (non-continual learning) and WDF (continual learning without dealing with forgetting), we see WDF is much better than NL for W2V. This indicates ASC tasks are similar and have shared knowledge. Catastrophic forgetting (CF) is not a major issue for W2V.

However, WDF is much worse than NL for BERT (with fine-tuning) and Adapter-BERT (with adapter-tuning). This is because BERT with fine-tuning learns highly task specific knowledge (Merchant et al., 2020). While this is desirable for NL, it is bad for WDF because task specific knowledge is hard to share across tasks or transfer. Then WDF causes serious forgetting (CF) for CL.

Model	Acc.	MF1
B-CL (-KSM;-TSM)	0.5403	0.4481
B-CL (-KSM)	0.8614	0.7852
B-CL (-TSM)	0.8312	0.7107
B-CL	**0.8829**	**0.8140**

Table 4: Ablation experiment results.

(3) Unlike BERT and Adapter-BERT, our B-CL can do very well in both forgetting avoidance and knowledge transfer (outperforming all baselines). For state-of-the-art CL baselines, EWC, UCL, OWM and HAT, although they perform better than WDF, they are all significantly poorer than B-CL as they don't have methods to encourage knowledge transfer. KAN and SRK do knowledge transfer but they are for document-level sentiment classification. They are weak, even weaker than other CL methods.

Effectiveness of Knowledge Transfer. We now look at knowledge transfer of B-CL. For forward transfer (B-CL(forward)) in Table 3), we use the test accuracy and MF1 of each task when it was first learned. For backward transfer (B-CL in Table 3), we use the final result after all tasks are learned. By comparing the results of NL with the results of forward transfer, we can see whether forward transfer is effective. By comparing the forward transfer result with the backward transfer result, we can see whether the backward transfer can improve further. The average results of B-CL forward (**B-CL(forward)**) and backward (**B-CL**) are given in Table 3. It shows that forward transfer of B-CL is highly effective (forward results for other CL baselines are given in the *Appendix* and we see B-CL's forward result outperforms all baselines' forward results). For backward transfer, B-CL slightly improves the performance.

Ablation Experiments. The results of ablation experiments are in Table 4. "-KSM;-TSM" means without knowledge sharing and task specific modules, simply deploying an Adapter-BERT. "-KSM" means without the knowledge sharing module. "-TSM" means without the task specific module. Table 4 clearly shows that the full B-CL system always gives the best overall results, indicating every component contributes to the model.

6 Conclusion

This paper studies continual learning (CL) of a sequence of ASC tasks. It proposed a novel technique called B-CL that can be applied to pre-trained BERT for CL. B-CL uses continual learning adapters and capsule networks to effectively encourage knowledge transfer among tasks and also to protect task-specific knowledge. Experiments show that B-CL markedly improves the ASC performance on both the new task and the old tasks via forward and backward knowledge transfer.

Acknowledgments

This work was supported in part by two grants from National Science Foundation: IIS-1910424 and IIS-1838770, a DARPA Contract HR001120C0023, and a research gift from Northrop Grumman.

References

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. 2019. Uncertainty-based continual learning with adaptive regularization. In *NIPS*.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient lifelong learning with A-GEM. In *ICLR*.

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

Zhiyuan Chen, Nianzu Ma, and Bing Liu. 2015. Lifelong learning for sentiment classification. In *ACL*.

Zhuang Chen and Tieyun Qian. 2019. Transfer capsule network for aspect level sentiment classification. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 international conference on web search and data mining*.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *LREC*.

Xu He and Herbert Jaeger. 2018. Overcoming catastrophic interference using conceptor-aided backpropagation. In *ICLR*.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *International conference on artificial neural networks*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *ICML*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD*.

Nitin Kamra, Umang Gupta, and Yan Liu. 2017. Deep generative dual memory network for continual learning. *CoRR*.

Zixuan Ke, Bing Liu, and Xingchang Huang. 2020a. Continual learning of a mixed sequence of similar and dissimilar tasks. In *NeurIPS*.

Zixuan Ke, Bing Liu, Hao Wang, and Lei Shu. 2020b. Continual learning with knowledge transfer for sentiment classification. In *ECML-PKDD*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*.

Matthias De Lange, Rahaf Aljundi, Marc Masana, and Tinne Tuytelaars. 2019. Continual learning: A comparative study on how to defy forgetting in classification tasks. *CoRR*.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NIPS*.

Bing Liu. 2015. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.

Qian Liu, Zhiqiang Gao, Bing Liu, and Yuanlin Zhang. 2015. Automated rule selection for aspect extraction in opinion mining. In *IJCAI*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *NIPS*.

Guangyi Lv, Shuai Wang, Bing Liu, Enhong Chen, and Kun Zhang. 2019. Sentiment classification by leveraging the shared knowledge from a sequence of domains. In *DASFAA*.

Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? *CoRR*.

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

Qi Qin, Wenpeng Hu, and Bing Liu. 2020. Using the past knowledge to improve sentiment classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1124–1133.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. icarl: Incremental classifier and representation learning. In *CVPR*.

Mohammad Rostami, Soheil Kolouri, and Praveen K. Pilly. 2019. Complementary learning for overcoming catastrophic forgetting using experience replay. In *IJCAI*.

Paul Ruvolo and Eric Eaton. 2013. ELLA: an efficient lifelong learning algorithm. In *ICML*, pages 507–515.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *NIPS*.

Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. 2017. Continual learning in generative adversarial nets. *CoRR*, abs/1705.08395.

Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *NIPS*.

Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning CRF for supervised aspect extraction. In *ACL*.

Daniel L. Silver, Qiang Yang, and Lianghao Li. 2013. Lifelong machine learning systems: Beyond learning algorithms. In *Lifelong Machine Learning, Papers from the 2013 AAAI Spring Symposium, Palo Alto, California, USA, March 25-27, 2013*.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *NAACL*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Hao Wang, Bing Liu, Shuai Wang, Nianzu Ma, and Yan Yang. 2019. Forward and backward knowledge transfer for sentiment classification. In *ACML*.

Shuai Wang, Guangyi Lv, Sahisnu Mazumder, Geli Fei, and Bing Liu. 2018. Lifelong learning memory networks for aspect sentiment classification. In *IEEE International Conference on Big Data*.

Rui Xia, Jie Jiang, and Huihui He. 2017. Distantly supervised lifelong learning for large-scale social media sentiment analysis. *IEEE Trans. Affective Computing*, 8(4):480–491.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL-HLT*.

Hu Xu, Sihong Xie, Lei Shu, and Philip S. Yu. 2018. Dual attention network for product compatibility and function satisfiability analysis. In *AAAI*.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. In *ACL*.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. 2019. Continuous learning of context-dependent processing in neural networks. *Nature Machine Intelligence*.

Wei Zhao, Haiyun Peng, Steffen Eger, Erik Cambria, and Min Yang. 2019. Towards scalable and reliable capsule networks for challenging NLP applications. In *ACL*.