CLASSIC: Continual and Contrastive Learning of Aspect Sentiment Classification Tasks

Zixuan Ke[1], **Bing Liu**[1], **Hu Xu**[2] and **Lei Shu**[3]*
[1]Department of Computer Science, University of Illinois at Chicago
[2]Facebook AI Research
[3]Amazon AWS AI
[1]{zke4,liub}@uic.edu
[2]huxu@fb.com
[3]shulindt@gmail.com

Abstract

This paper studies continual learning (CL) of a sequence of aspect sentiment classification (ASC) tasks in a particular CL setting called *domain incremental learning* (DIL). Each task is from a different domain or product. The DIL setting is particularly suited to ASC because in testing the system needs not know the task/domain to which the test data belongs. To our knowledge, this setting has not been studied before for ASC. This paper proposes a novel model called CLASSIC. The key novelty is a *contrastive continual learning* method that enables both knowledge transfer across tasks and knowledge distillation from old tasks to the new task, which eliminates the need for task ids in testing. Experimental results show the high effectiveness of CLASSIC.[1]

1 Introduction

Continual learning (CL) learns a sequence of tasks incrementally. After learning a task, its training data is often discarded (Chen and Liu, 2018). The CL setting is useful when the data privacy is a concern, i.e., the data owners do not want their data used by others (Ke et al., 2020b; Qin et al., 2020; Ke et al., 2021). In such cases, if we want to leverage the knowledge learned in the past to improve the new task learning, CL is appropriate as it shares only the learned model, but not the data. In our case, a task is a separate *aspect sentiment classification* (ASC) problem of a product or domain (e.g., camera or phone) (Liu, 2012). ASC is stated as follows: Given an aspect term (e.g., *sound quality* in a phone review) and a sentence containing the aspect (e.g., "*The sound quality is poor*"), ASC classifies whether the sentence expresses a positive, negative, or neutral opinion about the aspect.

There are three CL settings (van de Ven and Tolias, 2019): *Class Incremental Learning* (CIL), *Task Incremental Learning* (TIL), and *Domain Incremental Learning* (DIL). In CIL, the tasks contain non-overlapping classes. Only one model is built for all classes seen so far. In testing, no task information is provided. This setting is not suitable for ASC as ASC tasks have the same three classes. TIL builds one model for each task in a shared network. In testing, the system needs the task (e.g., phone domain) that each test instance (e.g., "*The sound quality is great*") belongs to and uses only the model for the task to classify the instance. Requiring the task information (e.g., phone domain) is a limitation. Ideally, the user should not have to provide this information for a test sentence. That is the DIL setting, i.e., all tasks sharing the same fixed classes (e.g., positive, negative, and neutral). In testing, no task information is required.

This work uses the DIL setting to learn a sequence of ASC tasks in a neural network. The key objective is to transfer knowledge across tasks to improve classification compared to learning each task separately. An important goal of any CL is to overcome *catastrophic forgetting* (CF) (McCloskey and Cohen, 1989), which means that in learning a new task, the system may change the parameters learned for previous tasks and cause their performance to degrade. We solve the CF problem as well; otherwise we cannot achieve improved accuracy. However, sharing the classification head for all tasks in DIL makes cross-task interfere/update inevitable. Without task information provided in testing makes DIL even more challenging.

Previous research has shown that one of the most effective approaches for ASC (Xu et al., 2019; Sun et al., 2019) is to fine-tune the BERT (Devlin et al., 2019) using the training data. However, our experiments show that this works poorly for DIL because the fine-tuned BERT on a task captures highly task specific features that are hard to use by other tasks.

In this paper, we propose a novel model called CLASSIC (**C**ontinual and contrastive **L**earning for

* Work was done prior to joining Amazon.

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

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6871–6883
November 7–11, 2021.

ASpect **S**ent**I**ment **C**lassification) in the DIL setting. Instead of fine-tuning BERT for each task, which causes serious CF, CLASSIC uses the idea of Adapter-BERT in (Houlsby et al., 2019) to avoid changing BERT parameters and yet achieve equally good results as BERT fine-tuning. A novel *contrative continual learning* method is proposed (1) to transfer the shareable knowledge across tasks to improve the accuracy of all tasks, and (2) to distill the knowledge (both shareable and not shareable) from previous tasks to the model of the new task so that *the new/last task model can perform all tasks*, which eliminates the need for task information (e.g., task id) in testing. Existing contrastive learning (Chen et al., 2020) cannot do these.

Task masks are also learned and used to protect task-specific knowledge to avoid forgetting (CF). Extensive experiments have been conducted to show the effectiveness of CLASSIC.

In summary, this paper makes the following contributions: **(1)** It proposes the problem of domain continual learning for ASC, which has not been attempted before. **(2)** It proposes a new model called CLASSIC that uses adapters to incorporate the pretrained BERT into the ASC continual learning, a novel *contrastive continual learning* method for knowledge transfer and distillation, and task masks to isolate task-specific knowledge to avoid CF.

2 Related Work

Several researchers have studied lifelong or continual learning for sentiment analysis. Early works are done under *Lifelong Learning* (LL) (Silver et al., 2013; Ruvolo and Eaton, 2013; Chen and Liu, 2014). Two Naive Bayes (NB) approaches were proposed to improve the new task learning (Chen et al., 2015; Wang et al., 2019). Xia et al. (2017) proposed a voting based approach. All these systems work on document sentiment classification (DSC). Shu et al. (2017) used LL for aspect extraction. These works do not use neural networks, and have no CF problem.

L2PG (Qin et al., 2020) uses a neural network but improves only the new task learning for DSC. Wang et al. (2018) worked on ASC, but since they improve only the new task learning, they did not deal with CF. Each task uses a separate network.

Existing CL systems SRK (Lv et al., 2019) and KAN (Ke et al., 2020b) are for DSC in the TIL setting, not for ASC. B-CL (Ke et al., 2021) is the first CL system for ASC. It also uses the idea of Adapter-BERT in (Houlsby et al., 2019) and is based on Capsule Network. More importantly, B-CL works in the TIL setting. The proposed CLASSIC system is based on contrastive learning and works in the DIL setting for ASC, which is a more realistic setting for practical applications.

General Continual Learning (CL): CL has been studied extensively in machine learning (Chen and Liu, 2018; Parisi et al., 2019). Existing work mainly focuses on dealing with CF. There are several main approaches. (1) *Regularization-based approaches* such as those in (Kirkpatrick et al., 2016; Lee et al.) add a regularization in the loss to consolidate previous knowledge when learning a new task. (2) *Parameter isolation-based approaches* such as those in (Serrà et al., 2018; Ke et al., 2020a; Abati et al., 2020) make different subsets of the model parameters dedicated to different tasks and identify and mask them out during the training of the new task. (3) *Replay-based approaches* 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, the generated data for previous tasks can help avoid CF.

These methods are for overcoming CF in the CIL or TIL setting of CL. Limited work has been done on knowledge transfer, which is our goal. There is little work in the DIL setting except the replay method DER++ (Buzzega et al., 2020), which saves some past data. CLASSIC saves no past data.

Contrastive learning (Chen et al., 2020; He et al., 2020) is the base of our contrastive continual learning method. However, there is a major difference. Existing contrastive learning uses various transformations (e.g., rotation and cropping) of the existing data (e.g., images) to generate different views of the data. However, we use the hidden space information from the previous task models to create views for explicit knowledge transfer and distillation. Existing contrastive learning cannot do that.

3 Proposed CLASSIC Method

State-of-the-art ASC systems all use BERT (Devlin et al., 2019) or other language models as the base. The proposed technique CLASSIC adopts the BERT-based ASC formulation in (Xu et al., 2019), where the aspect term (e.g., *sound quality*)

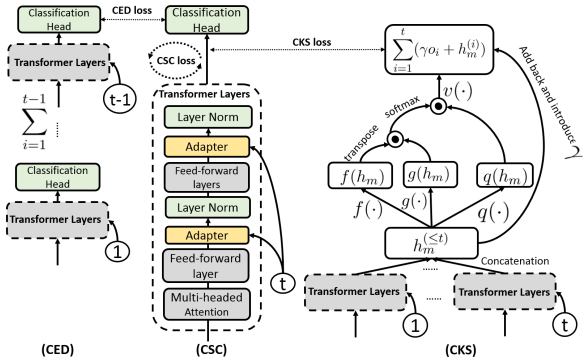

Figure 1: CLASSIC adopts Adapter-BERT (Houlsby et al., 2019) and its adapters (yellow boxes) in a transformer (Vaswani et al., 2017) layer (above (CSC)). An adapter is a 2-layer fully connected network with a skip-connection. It is added twice to each Transformer layer. Only the adapters and layer norm (green boxes) layers are trainable. The other modules (grey boxes) of BERT are frozen. **(CSC):** CSC loss is computed based on the current task model (details in Sec. 3.4). **(CED):** CED loss is computed based on all previous tasks from 1 to $t-1$ (details in Sec. 3.2). **(CKS):** CKS loss is computed based on previous and current tasks and a task-based self-attention. Details are given in Sec. 3.3.

and review sentence (e.g., "*The sound quality is great*") are concatenated via `[SEP]`. The sentiment polarity is predicted on top of the `[CLS]` token. As indicated earlier, although BERT can achieve state-of-the-art performance on a single task, its architecture and fine-tuning are unsuitable for CL (see Sec. 1) and perform very poorly (Sec. 4.4). We found that the BERT adapter idea in (Houlsby et al., 2019) is a better fit for CL.

BERT Adapter. The idea was given in Adapter-BERT (Houlsby et al., 2019), which inserts two 2-layer fully-connected networks (adapters) in each transformer layer of BERT (Figure 1(CSC)). During training for the end-task, only the adapters and normalization layers are updated. All the other BERT parameters are frozen. This is good for CL as fine-tuning the BERT causes serious forgetting. Adapter-BERT achieves similar accuracy to the fine-tuned BERT (Houlsby et al., 2019).

3.1 Overview of CLASSIC

The architecture of CLASSIC is given in Figure 1, which works in the DIL setting for ASC. It uses Adapter-BERT to avoid fine-tuning BERT. CLASSIC takes two inputs in training: (1) hidden states $h^{(t)}$ from the feed-forward layer of a transformer layer of BERT and (2) task id t (no task id is needed in testing, see Sec. 3.2.3). The outputs are hidden states with features for task t to build a classifier.

CLASSIC uses three sub-systems to achieve its objectives (see Sec. 1): (1) contrastive ensemble distillation (**CED**) for mitigating CF by distilling the knowledge of previous tasks to the current task model; (2) contrastive knowledge sharing (**CKS**) to encourage knowledge transfer; and (3) contrastive supervised learning on the current task model (**CSC**) to improve the current task model accuracy. We call this framework **contrastive continual learning**, inspired by contrastive learning.

Contrastive learning uses multiple views of the existing data for representation learning to group similar data together and push dissimilar data far away, which makes it easier to learn a more accurate classifier. It uses various transformations of the existing data to create useful views. Given a mini-batch of N training examples, if we create another view for each example, the batch will have $2N$ examples. We assume that i and j are two views of the training example. If we use i as ***the anchor***, (i, j) is called a positive pair. All other pairs (i, k) for $k \neq i$ are negative pairs. The contrastive loss for this positive pair is (Chen et al., 2020),

$$\mathcal{L}_{i,j} = -\log \frac{\exp((h_i \cdot h_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq j} \exp((h_i \cdot h_k)/\tau)}, \quad (1)$$

where the dot product $h_i \cdot h_j$ is regarded as a similarity function in the hidden space and τ is temperature. The final loss for the batch is calculated across all positive pairs. Eq. 1 is for unsupervised contrastive learning. It can also be used for supervised contrastive learning, where any two instances/views from the same class form a positive pair, and any instance of a class and any instance from other classes form a negative pair.

3.2 Overcoming Forgetting via Contrastive Ensemable Distillation (CED)

The CED objective is to deal with CF. We first introduce task masks that CED relies on to preserve the previous task knowledge/models to be distilled to the new task model to avoid CF.

3.2.1 Task Masks (TMs)

Given the input hidden states $h^{(t)}$ from the feed-forward layer of a transformer layer, the adapter maps them into input $k_l^{(t)}$ via a fully-connected network, where l is the l-th layer of the adapter. A TM (a "soft" binary mask) $\mathrm{m}_l^{(t)}$ is trained for each task t at each layer l in the adapter during training

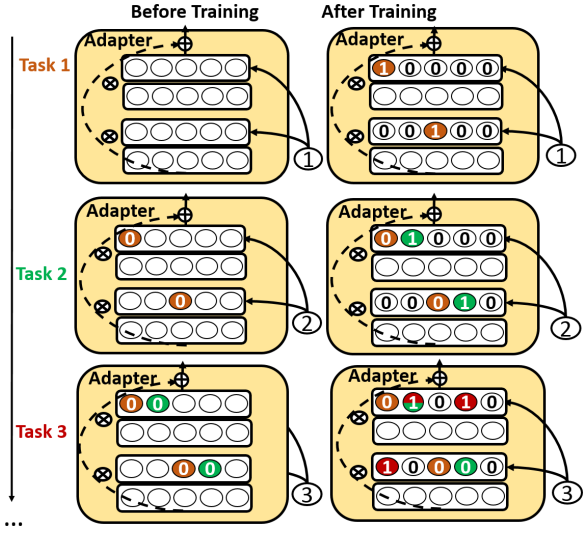

Figure 2: Illustration of task masking: a (learnable) task mask is applied after the activation function to *selectively* activate a neuron (or feature). The four rows of each task corresponds to the two fully-connected layers and their corresponding task masks. In the neurons before training, those with 0's are the neurons to be protected (masked) and those neurons without a number are free neurons (not used). In the neurons after training, those with 1's show neurons that are important for the current task, which are used as masks for the future. Those neurons with more than one color indicate that they are shared by more than one task. Those 0 neurons without a color are not used by any task.

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 train the TMs.

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

$$m_l^{(t)} = \sigma(s e_l^{(t)}). \tag{2}$$

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 the adapter, $k_l^{(t)}$, 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 2) After learning task t, the final $m_l^{(t)}$ is saved and added to the set $\{m_l^{(t)}\}$.

3.2.2 Training Task Masks (TMs)

For each previous task $i_{\text{prev}} \in \mathcal{T}_{\text{prev}}$, its TM $m_l^{(i_{\text{prev}})}$ 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 to 0. Before modifying the gradient, we first accumulate all used neurons by all previous tasks TMs. Since $m_l^{(i_{\text{prev}})}$ is binary, we use max-pooling to achieve the accumulation:

$$m_l^{(t_{\text{ac}})} = \text{MaxPool}(\{m_l^{(i_{\text{prev}})}\}). \tag{3}$$

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

$$g_l^{'(t)} = g_l^{(t)} \otimes (1 - m_l^{(t_{\text{ac}})}). \tag{4}$$

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. 2 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:

$$s = \frac{1}{s_{\max}} + (s_{\max} - \frac{1}{s_{\max}}) \frac{b-1}{B-1}, \tag{5}$$

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

Illustration. In Figure 2, after learning Task 1, 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 2, those useful neurons for Task 1 are masked (with "0" in those orange neurons on the left). The process also learns the useful neurons for Task 2 marked in green with "1"s. When Task 3 arrives, all neurons for Tasks 1 and 2 are masked, i.e., its TM entries are set to 0 (orange and green before training). After training Task 3, we see that Task 3 and Task 2 have a shared neuron that is important to both. The shared neuron is marked in both red and green.

3.2.3 Contrastive Ensemble Distillation (CED)

The TMs mechanism isolates different parameters for different tasks. This seems to be perfect for

overcoming forgetting since the previous task parameters are fixed and cannot be updated by future tasks. However, since DIL setting does not have task id in testing, we cannot directly take the advantage of the TMs. To address this issue, we propose the CED objective to help distill *all* previous knowledge to the current task model so that we can simply use *the last model* as the final model without requiring the task id in testing.

Representation of Previous Tasks. Recall that we know which neurons/units are for which task i by reading $\{m_l^{(i)}\}$. For each previous task i of the current task t, we can compute its masked output of Adapter-BERT $h_m^{(i)}$ (the layer before the classification head) by applying $m_l^{(i)}$ to the Adapter-BERT.

Ensemble Distillation Loss. We distill the knowledge of the ensemble of previous tasks into the single current task model. As we have a shared classification head for all tasks in DIL, which is exposed to forgetting, the distillation should be based on the output of the classification head. Specifically, given a previous task's Adapter-BERT output $h_m^{(i)}$, we compute the output of the classification head using $h_m^{(i)}$, which gives us the logit (unnormalized prediction) value $z_m^{(i)}$. We then distill the knowledge using $z_m^{(i)}$ and the current task classification head output $z_m^{(t)}$ based on contrastive loss, inspired by (Tian et al., 2020a),

$$\mathcal{L}_{\text{CED}}^{(i)} = \sum_{n=1}^{2N} -\log \frac{\exp((z_{m:2n-1}^{(i)} \cdot z_{m:2n}^{(t)})/\tau)}{\sum_{j=1}^{2N} \mathbb{1}_{n \neq j} \exp((z_{m:n}^{(i)} \cdot z_{m:j}^{(t)})/\tau)}, \tag{6}$$

where N is the batch size and $\tau > 0$ is an adjustable temperature parameter controlling the separation of classes. The index n is the *anchor* and the notation $z_{m:n}^{(i)}$ refers to the n-th sample in $z_m^{(i)}$. $z_{m:2n-1}^{(i)}$ and $z_{m:2n}^{(t)}$ are the logits of previous and current task models for the same input sample, a *positive* pair in contrastive learning. All the other possible pairs are *negative pairs*. Note that for each anchor i, there is 1 positive pair and $2N-2$ negative pairs. The denominator has a total of $2N-1$ terms (both the positives and negatives). Note that the previous task models are fixed and thus can serve as teacher networks. As we have $i \geq 1$ previous tasks, hence $i \geq 1$ teacher networks but only one current task student network. We adopt the contrastive framework by defining multiple pair-wise contrastive losses between $z_m^{(i)}$ and $z_m^{(t)}$. These losses are summed up to give the final CED loss,

$$\mathcal{L}_{\text{CED}} = \sum_{i=1}^{t-1} \mathcal{L}_{\text{CED}}^{(i)}. \tag{7}$$

3.3 Transferring Knowledge via Contrastive Knowledge Sharing (CKS)

CKS aims to capture the shared knowledge among tasks and help the new task learn a better representation and better classifier. The intuition of CKS is as follows: Contrasive learning has the ability to capture the shared knowledge between different views (Tian et al., 2020b; van den Oord et al., 2018). This is achieved by seeking representation that are invariant cross similar views. If we can generate a view from previous tasks that is similar to the current task, the contrastive loss can capture the shared knowledge and learn a representation for knowledge transfer to the new task learning. Below, we first introduce how to construct such a view and use it in the CKS objective.

3.3.1 Task-based Self-Attention

Intuitively, the more similar the two tasks are, the more shared knowledge they have. To achieve our goal, we should combine all similar tasks as the shared knowledge view. In order to focus on the similar tasks, we propose to use task-based self-attention mechanism to attend to them. Inspired by (Zhang et al., 2018), given the concatenation of the output of Adapter-BERT for all previous and current tasks, $h_m^{(\leq t)} = \text{cat}(\{h_m^{(i)}\}_{i=1}^{t})$, and task $i \leq t$, we first transform it into two feature spaces via $f(h_m^{(i)}) = W_f h_m^{(i)}$, $g(h_m^{(i)}) = W_g h_m^{(i)}$ (see Figure 1(CKS)).

To compare the similarity between tasks $i \leq t$ and $j \leq t$, we calculate similarity s_{ij} via

$$s_{ij} = f(h_m^{(i)})^T g(h_m^{(j)}). \tag{8}$$

We then compute the attention score $\alpha_{j,i}$ to indicate which similar tasks (similar to the current task t) should be attended to based on the current task data,

$$\alpha_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{t} \exp(s_{ij})}. \tag{9}$$

The attention score is applied to each task in $h_m^{(\leq t)}$ to get the attention output o_j using weighted sum:

$$o_j = v(\sum_{i=1}^{t} \alpha_{j,i} q(h_m^{(i)})), \tag{10}$$

where $v(\cdot)$ and $q(\cdot)$ are two functions for transforming feature spaces: $v(h_m^{(i)}) = W_v h_m^{(i)}$ and $q(h_m^{(i)}) = W_q h_m^{(i)}$.

Lastly, we multiply the output of the attention layer by a scale parameter and add back to the input feature $h_m^{(\leq t)}$. The final output of the $h_{\text{CKS}}^{(\leq t)}$ is the sum over all considered tasks,

$$h_{\text{CKS}}^{(\leq t)} = \sum_{i=1}^{t} (\gamma o_i + h_m^{(i)}), \tag{11}$$

where γ is a learnable scalar and it is initialized to 0. This allows the model to first learn on the current task and then gradually learn to assign more weights to other tasks.

3.3.2 Knowledge Sharing Loss

The output of the task-based self-attention provides us the knowledge sharing view $h_{\text{CKS}}^{(\leq t)}$. Along with the output of Adapter-BERT for the current task $h_m^{(t)}$, we can easily perform contrastive learning between these two views. Note that $h_{\text{CKS}}^{(\leq t)}$ is computed based on the current task data and their corresponding class labels, so we give the two views have the same label and thus we can integrate the label information in our CKS loss,

$$\mathcal{L}_{\text{CKS}} = \sum_{n=1}^{N} -\frac{1}{N_{y_n} - 1} \sum_{j=1}^{N} \mathbb{1}_{n \leq j} \mathbb{1}_{y_n = y_j} \log \frac{\exp((h_{\text{CKS}:n}^{(\leq t)} \cdot h_{m:j}^{(t)})/\tau)}{\sum_{k=1}^{N} \mathbb{1}_{n \neq k} \exp((h_{\text{CKS}:n}^{(\leq t)} \cdot h_{m:k}^{(t)})/\tau)}, \tag{12}$$

where N is the batch size and N_{y_n} is the number of examples in the batch that have the label y_n. $h_{\text{CKS}}^{(\leq t)}$ is the first view while $h_m^{(t)}$ is the second view. The shared knowledge between them represents the shared knowledge between previous and current tasks. Different from the CED loss, the CKS loss leverages the class information and thus can have multiple positive pairs decided by whether two samples share the same class label.

3.4 Contrastive Supervised Learning of the Current Task (CSC)

We further improve the performance of the current task by adopting the supervised contrastive loss (Khosla et al., 2020) on the *current task* $h_m^{(t)}$,

$$\mathcal{L}_{\text{CSC}} = \sum_{n=1}^{N} -\frac{1}{N_{y_n} - 1} \sum_{j=1}^{N} \mathbb{1}_{n \leq j} \mathbb{1}_{y_n = y_j} \log \frac{\exp((h_{m:n}^{(t)} \cdot h_{m:j}^{(t)})/\tau)}{\sum_{k=1}^{N} \mathbb{1}_{n \neq k} \exp((h_{m:n}^{(t)} \cdot h_{m:k}^{(t)})/\tau)}. \tag{13}$$

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 1: Number of sentences in each task or dataset. More detailed statistics are given in *Supplementary*.

3.5 Final Loss

The final loss is the weighted average of the supervised cross entropy (CE) loss, CSC loss, and the proposed CED and CKS losses:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{CSC}} + \lambda_2 \mathcal{L}_{\text{CED}} + \lambda_3 \mathcal{L}_{\text{CKS}}. \tag{14}$$

4 Experiments

This section evaluates the proposed CLASSIC system and compares it with both *non-continual learning* and *continual learning* baselines.

4.1 Experiment Datasets

We use 19 ASC datasets to produce sequences of 19 tasks. Each dataset is a set of aspect and sentiment annotated review sentences from reviews of a particular product and represents a task. The datasets are from 4 sources: (1) **HL5Domains** (Hu and Liu, 2004): review sentences of 5 products; (2) **Liu3Domains** (Liu et al., 2015): review sentences of 3 products; (3) **Ding9Domains** (Ding et al., 2008): review sentences of 9 products; and (4) **SemEval14**: review sentences of 2 products - SemEval 2014 Task 4 for laptop and restaurant. To be consistent with the existing research (Tang et al., 2016), sentences with both positive and negative sentiments about an aspect are not used. Statistics of the 19 datasets are given in Table 1.

4.2 Compared Baselines

We employ **46** baselines, which include both *non-continual learning* and *continual learning* methods. Since little work has been done in DIL, we adapt the recent TIL systems to DIL by merging classification heads to form DIL systems.

Non-Continual Learning Baselines: Each of these baselines builds a separate model for each task independently, which we call a ONE variant. It thus has no knowledge transfer or CF. There are 8 ONE variants. Four are created using **(1) BERT** with fine-tuning, (2) **BERT (Frozen)** without fine-tuning **(3) Adapter-BERT** (Houlsby et al., 2019) and **(4) W2V** (word2vec embeddings trained with the Amazon review data in (Xu et al., 2018) using FastText (Grave et al., 2018)). Adding CSC (Contrastive Supervised learning of the Current task) creates another 4 variants. We adopt the ASC network in (Xue and Li, 2018), taking aspect term and review sentence as input for BERT variants. For W2V variants, we use their concatenation.

Continual Learning (CL) Baselines. The CL setting has 38 baselines in 5 categories. The first category uses a naive CL (NCL) approach. It simply uses a network to learn all tasks with no mechanism to deal with CF or knowledge transfer. Like ONE, we have 8 NCL variants. The second category has 11 baselines created using recent CL methods **KAN** (Ke et al., 2020b), **SRK** (Lv et al., 2019), **HAT** (Serrà et al., 2018), **UCL** (Ahn et al., 2019), **EWC** (Kirkpatrick et al., 2016), **OWM** (Zeng et al., 2019) and **DER++** (Buzzega et al., 2020). KAN and SRK are for document sentiment classification. We use the concatenation of the aspect and the sentence as input. HAT, UCL, EWC, OWM and DER++ were originally designed for image classification. We replace their original image classification networks with CNN for text classification (Kim, 2014). HAT is one of the best TIL methods with almost no forgetting. UCL is a recent TIL method. EWC is a popular CIL method, which was adapted for TIL in (Serrà et al., 2018). They are converted to DIL versions by merging their classification heads. OWM (Zeng et al., 2019) is a CIL method, which we also adapt to a DIL method like EWC. DER++ and SRK can work in the DIL setting. HAT and KAN require task id as an input in testing and cannot function in the DIL setting. We create two variants of HAT (and KAN): using the *last* model in testing as CLASSIC does or detecting task id using the entropy method *ent* in (von Oswald et al., 2020). This category uses BERT (Frozen) as the base. The third category has 7 baselines using Adapter-BERT. KAN and SRK cannot be adapted to use adapters. The fourth category uses W2V, which gives another 11 baselines. The final category has one baseline **LAMOL** (Sun et al., 2020), which uses the GPT-2 model.

Evaluation Protocol: We follow the standard CL evaluation method in (Lange et al., 2019). We first present CLASSIC a sequence of ASC tasks for it to learn. Once a task is learned, its training data is discarded. After all tasks are learned, we test using the test data of all tasks without giving task ids.

4.3 Hyperparameters

Unless otherwise stated, the adapter uses 2 layers of fully connected network with dimensions 2000. The task id embeddings have 2000 dimensions. A fully connected layer with softmax output is used as the classification head in the last layer of BERT. We use 400 for $s_{\max}$ in Eq. 5, dropout of 0.5 between fully connected layers. The temperature τ in each contrastive objective is set to 1 (see Supplementary for parameter tuning). The weight of each objective in Eq. 14 is set to 1. We use the embedding of `[CLS]` as the output of Adapter-BERT. For CKS and CSC, we use l_2 normalization on the output of Adapter-BERT before computing the contrastive loss. The training of BERT, Adapter-BERT and CLASSIC follow that of (Xu et al., 2019). We adopt $\text{BERT}_{\textbf{BASE}}$ (uncased). The max length of the sum of sentence and aspect is 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 CL baselines, we train all models with the learning rate of 0.05, early-stop training when there is no improvement in the validation loss for 5 epochs and set the batch size to 64. We use the code provided by their authors and adopt their original parameters (for EWC, we adopt the variant implemented by (Serrà et al., 2018)).

4.4 Results and Analysis

As the order of the 19 tasks can influence the final results, we randomly select and run 5 task sequences and report their average results in Table 2. We compute both accuracy and Macro-F1, where Macro-F1 is the main metric as the imbalanced classes introduce biases in accuracy.

Scenario	Category	Model	Acc.	MF1
Non-continual Learning	BERT	ONE	0.8584	0.7635
		ONE+csc	0.8353	0.7388
	BERT (Frozen)	ONE	0.7814	0.5813
		ONE+csc	0.8265	0.7232
	Adapter-BERT	ONE	0.8596	0.7807
		ONE+csc	0.8530	0.7516
	W2V	ONE	0.7701	0.5189
		ONE+csc	0.7761	0.5487
Continual Learning	BERT	NCL	0.8048	0.7085
		NCL+csc	0.7727	0.5807
	BERT (Frozen)	NCL	0.8685	0.7873
		NCL+csc	0.8693	0.7912
	Adapter-BERT	NCL	0.8667	0.7804
		NCL+csc	0.8809	0.7847
	W2V	NCL	0.8408	0.7455
		NCL+csc	0.8396	0.7509
	BERT (Frozen)	KAN	—	
		KAN+last	0.8320	0.7352
		KAN+ent	0.8278	0.7243
		SRK	0.8391	0.7438
		EWC	0.8660	0.7831
		UCL	0.8538	0.7690
		OWM	0.8611	0.7665
		DER++	0.8753	0.8009
		HAT	—	
		HAT+last	0.8473	0.7649
		HAT+ent	0.8418	0.7614
	Adapter-BERT	EWC	0.8805	0.7875
		UCL	0.7123	0.3961
		OWM	0.8766	0.7882
		DER++	0.8859	0.7985
		HAT	—	
		HAT+last	0.8823	0.7919
		HAT+ent	0.8854	0.8245
	W2V	KAN	—	
		KAN+last *	0.7123	0.3961
		KAN+ent *	0.7123	0.3961
		SRK *	0.7123	0.3961
		EWC	0.7586	0.6545
		UCL	0.8187	0.6965
		OWM	0.8256	0.7253
		DER++	0.8459	0.7722
		HAT	—	
		HAT+last	0.7599	0.5849
		HAT+ent	0.7605	0.5349
	LAMOL		0.8891	0.8059
	CLASSIC (forward)		0.8886	0.8365
	CLASSIC		**0.9022**	**0.8512**

Table 2: Accuracy (Acc.) and Macro-F1 (MF1) averaged over 5 random sequences of 19 tasks. KAN and HAT need task id in testing and thus have no results. KAN and SRK (RNN based) cannot work with Adapters. *: KAN and SRK under W2V fail to train. **Standard deviation** showing statistical significance, **network size** and **running time** are in Supplementary.

Overall, Table 2 shows that CLASSIC outperforms all baselines markedly.

(1). For non-continual learning baselines (ONE variants), Adapter-BERT performs similarly to BERT (fine-tuning). Both BERT (Frozen) and W2V variants are weaker, which is understandable.

(2). Comparing ONE variants and NCL variants, we see that under W2V, NCL variants are much better than ONE variants. This indicates ASC tasks are similar and have shared knowledge. Catastrophic forgetting (CF) is not a major issue for W2V.

However, BERT NCL (fine-tuning) is much worse than BERT ONE and Adapter-BERT NCL (adapter-tuning) as BERT fine-tuning learns highly task specific knowledge (Merchant et al., 2020). While this is desirable for ONE, it is bad for NCL because task specific knowledge is hard to share across tasks, which causes forgetting (CF). The +csc options are poor for BERT ONE and NCL.

(3). Various continual learning (CL) baselines with BERT (Frozen) are also markedly weaker than CLASSIC. Baselines that can use Adapter-BERT are also much poorer than CLASSIC. Note that SRK and KAN cannot work with Adapter-BERT.

(4). W2V based CL baselines are even weaker.

(5). Since both KAN and HAT need task id in testing and the DIL setting does not provide task id, they have no results. But we use the last model (+last) or use an existing entropy-based method (+ent) (von Oswald et al., 2020) to automatically identify the task id for each test instance. These variants are also markedly weaker than CLASSIC.

(6). LAMOL is based on GPT-2 and its performance is weaker than CLASSIC too.

Effectiveness of Knowledge Transfer. The results under CLASSIC(forward) in Table 2 are the average results computed using the accuracy/MF1 of each task when it was first learned. The results under CLASSIC are the final average results after all tasks are learned, including backward transfer. By comparing ONE variants and CLASSIC(forward), we can see whether forward transfer is effective. By comparing CLASSIC(forward) and CLASSIC, we can see whether the backward transfer can improve further. We see both forward and backward transfers are effective.

4.5 Ablation Experiments

The results of ablation experiments are given in Table 3. "-CKS", "-CSC" and "-CED" mean without constrastive knowledge sharing, contrastive supervised learning on the current task and contrastive ensemble distillation, respectively. Table 3 clearly shows that each of the components is effective and they work in concert to produce the best final result.

5 Conclusion

This paper studied *domain incremental learning* (DIL) of a sequence of ASC tasks without knowing

Model	Acc.	MF1
CLASSIC	**0.9022**	**0.8512**
-CSC	0.8872	0.8007
-CKS	0.8915	0.8232
-CED	0.8828	0.7934
-CKS,-CED	0.8864	0.7969
-CKS,-CSC	0.8926	0.8346
-CED,-CSC	0.8868	0.8032
-CED,-CKS,-CSC	0.8823	0.7919

Table 3: Ablation experiment results.

the task ids in testing. Our method CLASSIC uses *adapters* to exploit BERT and to deal with BERT CF in fine-tuning, and the proposed *contrastive continual learning* to transfer knowledge across tasks and to distill knowledge from previous tasks to the current task so that the last model can be used for all tasks in testing and no task id is needed. Our experimental results show that CLASSIC outperforms the state-of-the-art baselines.

Finally, we believe that the idea of CLASSIC is also applicable to some other NLP tasks. For example, in named entity extraction, we can build a better model to extract the same types of entities from text of different domains. Each domain works on the same task but no data sharing (the data may be from different clients with privacy concerns). Since this is an extraction task, the backbone model needs to be switched to an extraction model.

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

Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. 2020. Conditional channel gated networks for task-aware continual learning. In *CVPR-2020*, pages 3931–3940.

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

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. *arXiv preprint arXiv:2004.07211*.

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

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Zhiyuan Chen and Bing Liu. 2014. Topic modeling using topics from many domains, lifelong learning and big data. In *ICML*.

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*.

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*.

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

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

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

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*.

Zixuan Ke, Hu Xu, and Bing Liu. 2021. Adapting bert for continual learning of a sequence of aspect sentiment classification tasks. In *NAACL*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning.

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. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

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*.

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.

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*.

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

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

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020a. Contrastive representation distillation. In *ICLR*.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. 2020b. What makes for good views for contrastive learning? In *NeurIPS 2020*.

Gido M. van de Ven and Andreas S. Tolias. 2019. Three scenarios for continual learning. *https://arxiv.org/abs/1904.07734*.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*.

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*.

Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. 2020. Continual learning with hypernetworks. In *ICLR*.

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*.

Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. 2018. Self-attention generative adversarial networks.

A Appendix

A.1 Detailed Datasets Statistics

Table 1 in the main paper has already showed the number of examples in each dataset. Here we provide additional details about aspects and sentiments. The detailed statistics of the 19 datasets or tasks are given in Table 4 here.

A.2 Standard Deviations

Table 5 reports the standard deviation of CLASSIC and the considered baselines over 5 runs with random seeds using one random task sequence. We can see the results are stable.

A.3 CLASSIC in TIL Scenario

Table 6 shows that our proposed method CLASSIC can be adapted for the *Task Incremental Learning* (TIL) setting of continual learning, which requires the task ids during testing, but our DIL setting does not require. We can observe that CLASSIC in the TIL setting also outperforms existing TIL baselines.

A.4 Execution Time and Number of Parameters

Table 7 reports the number of parameters (regardless of trainable or non-trainable) and the training execution times of different models. The execution time is computed as the average training time *per* task. Our experiments were run on GeForce GTX 2080 Ti with 11G GPU memory.

A.5 Hyper-parameters and Validation Results

Sec. 4.3 in the main paper reported the best hyper-parameters. Regarding hyper-parameter search, we performed grid search on the *temperature* parameter τ within {0.03, 0.5, 0.8, 1}, *batch size* within {32, 64, 128}, and s_{max} within {140, 200, 300, 400}. We also experimented with whether to apply the l_2 normalization before contrast and whether to use the *logits* or the *second last* layer to do the contrast. We did not save the validation results but the reported test results in the paper are given by the parameters with the best validation performance. We encourage the reviewers and interested readers to play with the submitted code.

Dataset	Domains	Training	Validating	Testing
Liu3domain	Speaker	233 S./352 A./287 P./65 N./0 Ne.	30 S./44 A./35 P./9 N./0 Ne.	38 S./44 A./40 P./4 N./0 Ne.
	Router	200 S./245 A./142 P./103 N./0 Ne.	24 S./31 A./19 P./12 N./0 Ne.	22 S./31 A./24 P./7 N./0 Ne.
	Computer	187 S./283 A./218 P./65 N./0 Ne.	25 S./35 A./23 P./12 N./0 Ne.	29 S./36 A./29 P./7 N./0 Ne.
HL5domain	Nokia6610	209 S./271 A./198 P./73 N./0 Ne.	29 S./34 A./30 P./4 N./0 Ne.	28 S./34 A./25 P./9 N./0 Ne.
	Nikon4300	131 S./162 A./135 P./27 N./0 Ne.	15 S./20 A./18 P./2 N./0 Ne.	15 S./21 A./19 P./2 N./0 Ne.
	Creative	582 S./677 A./422 P./255 N./0 Ne.	68 S./85 A./42 P./43 N./0 Ne.	70 S./85 A./52 P./33 N./0 Ne.
	CanonG3	190 S./228 A./180 P./48 N./0 Ne.	25 S./29 A./21 P./8 N./0 Ne.	24 S./29 A./24 P./5 N./0 Ne.
	ApexAD	281 S./343 A./146 P./197 N./0 Ne.	35 S./43 A./16 P./27 N./0 Ne.	28 S./43 A./31 P./12 N./0 Ne.
Ding9domain	CanonD500	103 S./118 A./96 P./22 N./0 Ne.	11 S./15 A./14 P./1 N./0 Ne.	13 S./15 A./11 P./4 N./0 Ne.
	Canon100	137 S./175 A./123 P./52 N./0 Ne.	19 S./22 A./20 P./2 N./0 Ne.	16 S./22 A./21 P./1 N./0 Ne.
	Diaper	166 S./191 A./143 P./48 N./0 Ne.	22 S./24 A./18 P./6 N./0 Ne.	24 S./24 A./22 P./2 N./0 Ne.
	Hitachi	152 S./212 A./153 P./59 N./0 Ne.	23 S./26 A./19 P./7 N./0 Ne.	23 S./27 A./14 P./13 N./0 Ne.
	Ipod	124 S./153 A./101 P./52 N./0 Ne.	18 S./19 A./14 P./5 N./0 Ne.	19 S./20 A./15 P./5 N./0 Ne.
	Linksys	152 S./176 A./128 P./48 N./0 Ne.	19 S./22 A./13 P./9 N./0 Ne.	20 S./23 A./16 P./7 N./0 Ne.
	MicroMP3	384 S./484 A./340 P./144 N./0 Ne.	42 S./61 A./48 P./13 N./0 Ne.	51 S./61 A./39 P./22 N./0 Ne.
	Nokia6600	298 S./362 A./244 P./118 N./0 Ne.	26 S./45 A./32 P./13 N./0 Ne.	39 S./46 A./30 P./16 N./0 Ne.
	Norton	168 S./194 A./54 P./140 N./0 Ne.	17 S./24 A./15 P./9 N./0 Ne.	24 S./25 A./5 P./20 N./0 Ne.
SemEval14	Rest	1893 S./3452 A./2094 P./779 N./579 Ne.	84 S./150 A./70 P./26 N./54 Ne.	600 S./1120 A./728 P./196 N./196 Ne.
	Laptop	1360 S./2163 A./930 P./800 N./433 Ne.	98 S./150 A./57 P./66 N./27 Ne.	411 S./638 A./341 P./128 N./169 Ne.

Table 4: Statistics of the datasets. **S**.: number of sentences; **A**: number of aspects; **P., N., and Ne.**: number of positive, negative and neutral aspect polarities respectively. Note that SemEval14 has 3 classes of polarities while the others have only 2 classes (positive and negative) because in these datasets, those sentences without sentiment (neutral) are not annotated with aspects. Thus, we cannot use them for aspect sentiment classification (ASC).

Scenario	Category	Model	DIL Acc.	DIL MF1
Non-continual Learning	BERT	ONE	±0.0145	±0.0300
		ONE+csc	±0.0127	±0.0336
	BERT (Frozen)	ONE	±0.0100	±0.0024
		ONE+csc	±0.0140	±0.0149
	Adapter-BERT	ONE	±0.0170	±0.0379
		ONE+csc	±0.0094	±0.0327
	W2V	ONE	±0.0129	±0.0206
		ONE+csc	±0.0092	±0.0079
Continual Learning	BERT	NCL	±0.0137	±0.0228
		NCL+csc	±0.0266	±0.0328
	BERT (Frozen)	NCL	±0.0065	±0.0110
		NCL+csc	±0.0085	±0.0116
	Adapter-BERT	NCL	±0.0102	±0.0128
		NCL+csc	±0.0102	±0.0130
	W2V	NCL	±0.0192	±0.0230
		NCL+csc	±0.0121	±0.0108
	BERT (Frozen)	KAN	—	
		KAN+last	±0.0032	±0.0045
		KAN+ent	±0.0056	±0.0065
		SRK	±0.0069	±0.0099
		EWC	±0.0094	±0.0093
		UCL	±0.0084	±0.0100
		OWM	±0.0092	±0.0140
		DER++	±0.0096	±0.0120
		HAT	—	
		HAT+last	±0.0095	±0.0098
		HAT+ent	±0.0029	±0.0097
	Adapter-BERT	EWC	±0.0110	±0.0110
		UCL	±0.0000	±0.0000
		OWM	±0.0052	±0.0109
		DER++	±0.0137	±0.0179
		HAT	—	
		HAT+last	±0.0038	±0.0055
		HAT+ent	±0.0059	±0.0106
	W2V	KAN	—	
		KAN+last	±0.0000	±0.0000
		KAN+ent	±0.0000	±0.0000
		SRK	±0.0000	±0.0000
		EWC	±0.0059	±0.0076
		UCL	±0.0097	±0.0128
		OWM	±0.0077	±0.0081
		DER++	±0.0075	±0.0041
		HAT	—	
		HAT+last	±0.0053	±0.0082
		HAT+ent	±0.0103	±0.0199
	LAMOL		±0.0027	±0.0062
	CLASSIC		±0.0048	±0.0101

Table 5: Standard deviations. HAT and KAN have no results because they require task ids in testing, but in the DIL setting, no task ids are provided in testing.

Scenario	Category	Model	TIL Acc.	TIL MF1	DIL Acc.	DIL MF1
Non-continual Learning	BERT	ONE	0.8584	0.7635	0.8584	0.7635
		ONE+csc	0.8353	0.7388	0.8353	0.7388
	BERT (Frozen)	ONE	0.7814	0.5813	0.7814	0.5813
		ONE+csc	0.8265	0.7232	0.8265	0.7232
	Adapter-BERT	ONE	0.8596	0.7807	0.8596	0.7807
		ONE+csc	0.8530	0.7516	0.8530	0.7516
	W2V	ONE	0.7701	0.5189	0.7701	0.5189
		ONE+csc	0.7761	0.5487	0.7761	0.5487
Continual Learning	BERT	NCL	0.4960	0.4308	0.8048	0.7085
		NCL+csc	0.5939	0.3416	0.7727	0.5807
	BERT (Frozen)	NCL	0.8551	0.7664	0.8685	0.7873
		NCL+csc	0.8783	0.8271	0.8693	0.7912
	Adapter-BERT	NCL	0.5403	0.4481	0.8667	0.7804
		NCL+csc	0.8630	0.8090	0.8809	0.7847
	W2V	NCL	0.8269	0.7356	0.7736	0.6765
		NCL+csc	0.8421	0.7418	0.8396	0.7509
	BERT (Frozen)	KAN	0.8549	0.7738	—	
		KAN+last	—		0.8320	0.7352
		KAN+ent	—		0.8278	0.7243
		SRK	0.8476	0.7852	0.8391	0.7438
		EWC	0.8637	0.7452	0.8660	0.7831
		UCL	0.8389	0.7482	0.8538	0.7690
		OWM	0.8702	0.7931	0.8611	0.7665
		DER++	0.8427	0.7508	0.8753	0.8009
		HAT	0.8674	0.7816	—	
		HAT+last	—		0.8473	0.7649
		HAT+ent	—		0.8418	0.7614
	Adapter-BERT	EWC	0.5630	0.4958	0.8805	0.7875
		UCL	0.6446	0.3664	0.7123	0.3961
		OWM	0.7299	0.6651	0.8766	0.7882
		DER++	0.4763	0.3554	0.8859	0.7985
		HAT	0.8614	0.7852	—	
		HAT+last	—		0.8823	0.7919
		HAT+ent	—		0.8854	0.8245
	W2V	KAN	0.7206	0.4001	—	
		KAN+last	—		0.7123	0.3961
		KAN+ent	—		0.7123	0.3961
		SRK	0.7101	0.3963	0.7123	0.3961
		EWC	0.8416	0.7229	0.7586	0.6545
		UCL	0.8441	0.7599	0.8187	0.6965
		OWM	0.8270	0.7118	0.8256	0.7253
		DER++	0.8327	0.6993	0.8459	0.7722
		HAT	0.8083	0.6363	—	
		HAT+last	—		0.7599	0.5849
		HAT+ent	—		0.7605	0.5349
	LAMOL		0.8891	0.8059	0.8891	0.8059
	CLASSIC (forward)		**0.8897**	**0.8338**	**0.8886**	**0.8365**
	CLASSIC		**0.8942**	**0.8393**	**0.9022**	**0.8512**

Table 6: Accuracy (Acc.) and Macro-F1 (MF1) averaged over 5 random sequences of 19 tasks.

Scenarios	Category	Model	#parameters (M)	Running time (s)
Non-continual Learning	BERT	ONE	109.5	600.0
	BERT (Frozen)	ONE	110.4	500.0
	Adapter-BERT	ONE	183.3	684.0
	W2V	ONE	6.7	189.0
Continual Learning	BERT	NCL	109.5	600.0
	BERT (Frozen)	NCL	110.4	500.0
	Adapter-BERT	NCL	183.3	684.0
	W2V	NCL	6.7	189.0
	BERT (frozen)	KAN	116.6	550.0
		SRK	117.8	600.0
		EWC	110.4	580.0
		UCL	110.4	638.0
		OWM	110.6	635.0
		DER++	115.0	650.0
		HAT	111.3	610.0
	Adapter-BERT	EWC	183.3	840.0
		UCL	183.4	870.0
		OWM	184.4	780.0
		DER++	184.0	880.0
		HAT	185.2	840.0
	W2V	KAN	7.0	150.0
		SRK	7.2	160.0
		EWC	6.2	162.0
		UCL	6.2	135.0
		OWM	6.4	125.0
		DER++	6.8	135.0
		HAT	6.4	180.0
	LAMOL		124.4	650.0
	CLASSIC		185.2	900.0

Table 7: Network size (number of parameters, regardless of trainable or non-trainable) and average training execution time per task of each model in seconds.