Continual Learning with Transformers for Image Classification

Beyza Ermis
AWS, Berlin
ermibeyz@amazon.com

Giovanni Zappella
AWS, Berlin
zappella@amazon.com

Martin Wistuba
AWS, Berlin
marwistu@amazon.com

Aditya Rawal
AWS, Santa Clara
adirawal@amazon.com

Cédric Archambeau
AWS, Berlin
cedrica@amazon.com

Abstract

In many real-world scenarios, data to train machine learning models become available over time. However, neural network models struggle to continually learn new concepts without forgetting what has been learnt in the past. This phenomenon is known as catastrophic forgetting and it is often difficult to prevent due to practical constraints, such as the amount of data that can be stored or the limited computation sources that can be used. Moreover, training large neural networks, such as Transformers, from scratch is very costly and requires a vast amount of training data, which might not be available in the application domain of interest. A recent trend indicates that dynamic architectures based on an expansion of the parameters can reduce catastrophic forgetting efficiently in continual learning, but this needs complex tuning to balance the growing number of parameters and barely share any information across tasks. As a result, they struggle to scale to a large number of tasks without significant overhead. In this paper, we validate in the computer vision domain a recent solution called Adaptive Distillation of Adapters (ADA), which is developed to perform continual learning using pre-trained Transformers and Adapters on text classification tasks. We empirically demonstrate on different classification tasks that this method maintains a good predictive performance without retraining the model or increasing the number of model parameters over the time. Besides it is significantly faster at inference time compared to the state-of-the-art methods.

1. Introduction

The ability to learn from evolving streams of training data is important for many real-world applications. While neural networks showed a great ability to learn a task, but when confronted with a sequence of different ones they tend to override the previous concepts. Deep networks suffer heavily from this phenomenon called *catastrophic forgetting* (CF) [21], impeding continual or lifelong learning. A growing amount of efforts have emerged to tackle catastrophic forgetting [7, 11, 16, 26, 33, 34] in continual learning (CL). Existing methods can be roughly categorized as replay-based methods [3, 4, 20, 27, 31] that retain some training data of old tasks and use them in learning a new task to circumvent the issue of CF; regularization-based methods [1, 14, 16] add a regularization term to the loss to consolidate previous knowledge when learning a new task; and parameter isolation methods that can dynamically expand the network architectures [19, 36] or re-arrange their structures [10, 15]. Replay-based methods explicitly retrain on a subset of stored samples while training on new tasks. That can induce dramatic memory overhead when tackling a large number of tasks or regulatory-related issues due to the data storage. Regularization-based methods are often difficult to tune due to the sensitivity to the importance of the regularization term. In practice, this complexity often leads to poor performance. Parameter-isolation methods either keep augmenting additional parameters with the consequence of significantly increasing the number of parameters, or need complex pruning as post-processing with requirement to know which parameters should be kept/pruned.

Inspired by the significant achievement of Transformers [30] in Natural Language Processing (NLP), some pioneering works have recently been done on adapting transformer architectures to Computer Vision (CV). Vision Transformer (ViT) [6] showed that a pure Transformer applied directly to a sequence of image patches can perform well on image classification tasks if the training dataset is sufficiently large. Data-Efficient Image Transformers (DeiT) [29] further demonstrated that Transformers can be successfully trained on standard datasets, such as ImageNet-1K [5]. Besides, some recent studies [2, 18, 35] showed transformers have a good ability to generalize well

to new domains with a few samples. To this end, *we leverage the vision transformers* to improve the ease of use of CL frameworks for real-world applications. To the best of our knowledge, only a few recent works [8, 18] have applied the transformer architecture to CL on image datasets, but they also require rehearsal memory or training a new transformer model from scratch (cannot work with pre-trained transformer models).

Due to the drawbacks that we listed above, existing methods do not seem to fit the requirements of many practical CL applications. We found the work of [9] as a promising candidate to adopt since it does not require re-training the model nor storing the old task data, keep the memory and resources consumption limited (e.g. constant number of parameters) as the number of tasks grows and can work with pre-trained transformer models. This method stores a small set of covariates to be used for distillation, but given the limited size and usage, they can easily be replaced with external data sources (e.g., data provided by the owner of the model and not subject to the same legal restrictions of user data). It introduces a reasonable amount of additional parameters (about 12% of the BERT size in their experiments) and only adds a new linear head for every task. Moreover, it is tested with several pre-trained transformer models for text (i.e., BERT, DistillBERT and RoBERTa). Nevertheless, there is no evidence that this may work for CV Transformers. Our main focus is to answer the question: Can this technique provide good performance also for CV?

In this study, we address an image classification problem on a sequence of classification tasks provided in sequence using pre-trained models: ViT and DeiT. While the solution can be used in several different tasks, we limit our study to image classification since image classification is core to computer vision and it is often used as a benchmark to measure progress in image understanding. We conjecture that the behaviour observed in these experiments will be informative also for related tasks such as detection or segmentation.

We test the efficiency of Adaptive Distillation of Adapters (ADA) on CIFAR100 and MiniImageNet. Our main contributions are: 1) Using Adapters approach with vision transformers for the first time on continual image classification tasks 2) Validate that Adapters work with vision transformers and show that Adaptive Distillation of Adapters (ADA) can achieve predictive performance on-par with memory-hungry methods such AdapterFusion [24], 3) Adding a benchmark with CL methods such Elastic Weight Consolidation (EWC) [16] and Experience Replay (ER) [27] by using transformer architectures.

Related Work. We discuss the related work in two folds: Adapters and CL with vision transformers. Residual adapters [25] adds a few learnable residual layers to the standard ResNet model and train only these newly added layers, has been proposed as a transfer learning technique used for multi-domain learning and domain adaptation. These adapters are designed specifically for ResNet models, so they are not generalizable to other networks. Recently Houlsby *et al.* [12] proposed Adapters for Transformers in NLP to isolate task-specific knowledge for multi-task learning. However, in the sequential learning setting, Adapters keep increasing the model parameters with each task, so the memory requirements. In the CL direction, [9] uses Adapters for text classification, but by using distillation after training a set of task, it manages to keep the memory size constant while maintaining a good prediction performance for both old and new tasks. Benchmarking this approach on CV tasks is the main focus of our work.

Two recent works are related with the method we are testing. In [18], for each task, before training on a new task, the model is copied and fixed to be used as the teacher model in the distillation phase. The student model is trained on both new task samples together with the knowledge distillation loss that uses samples from old tasks which is stored in the rehearsal memory. In [8], they aim to learn a unified model that will classify an increasingly growing number of classes by building upon a new architecture. However, they need to train a new transformer model, where the process is very costly. Our main goal is to use public pre-trained models.

2. Problem setup

In this section we formalize our goal of CL on a sequence of image classification tasks $\{T_1, \ldots, T_N\}$ where each task T_i contains a different set of image-label training pairs $(x^i_{1:t}, y^i_{1:t})$. Each task T_i may contain c new classes namely $Y_i = \{Y_i^1, \ldots, Y_i^c\}$ and each new class has t examples. T_i is sampled iid from a distribution $D_i(X_i, Y_i)$. Each task represents a different classification problem and the learner creates a new classification head for each. The task identifier is provided also at inference time.

The goal of the learner is to learn a set of parameters $\tilde{\Theta}$ such that $\frac{1}{N}\sum_{i\in\{1,\ldots,N\}} \text{loss}(T_i; \tilde{\Theta})$ is minimized. In our specific case, $\tilde{\Theta}$ is composed of a set of parameters Θ provided by a pre-trained model and, depending on the algorithm, additional network parts whose weights need to be learnt. In its simplest case this additional set of model parameters can just be a head model but, some algorithms use significantly more elaborate functions.

For the training of task T_i, the algorithm can access the data provided in the current batch and some data which eventually stored in memory. To evaluate the system, the test data consists of examples across all the previous tasks. All methods in the following sections receive as input a pre-trained language model $f_\Theta(.)$, e.g., ViT [6], parameterized

by Θ. The pre-trained model receives an input image, called x_i and it is able to compute a representation for it.

3. The ADA algorithm

In this section we provide an overview of the main components used by Adaptive Distillation of Adapters (ADA) and the algorithm itself. ADA is performed in two steps: the first step is to train an adapter model and a new classifier using the new task T_i's training dataset D_i, which is referred as the *new* model; the second step is to consolidate the *old* model(s), the model(s) obtained in the previous round, and new model.

Adapters. Adapters share a large set of parameters Θ across all tasks and introduce a small number of task-specific parameters Φ_i. Current work on adapters focuses on training adapters for each task separately. For each of the N tasks, the model is initialized with parameters of a pre-trained model Θ. In addition, a set of new and randomly initialized adapter parameters Φ_i are introduced for tasks $i \in \{1, \ldots, N\}$. The parameters Θ are fixed and only the parameters Φ_i are trained. This makes it possible to train adapters for all N tasks, and store the corresponding knowledge in designated parts of the model. The objective for each task $i \in \{1, \ldots, N\}$ is of the form: $\Phi_i \leftarrow \arg\min_{\Phi} L_i(D_i; \Theta, \Phi)$.
In this work, we propose to define and use adapters for vision transformers [6, 29] While this provides good predictive performance, in the CL setting, new tasks are added sequentially and storing a large set of adapters $\Phi_1, \ldots, \Phi_N$ is practically infeasible. As in AdapterBERT, we insert a 2-layer fully-connected network in each transformer layer of ViT and DeiT (see Figure 1). DeiT is built upon the ViT architecture, so the Adapter is added in the same way.

Distillation of Adapters. For each new task T_i, the adapter parameters Φ_i are added to the model, while the pre-trained model parameters Θ are kept frozen. Only the task-specific model parameters Φ_i and the head model parameters h_i are trained for the current task. The model $f_i(x; \Theta, \Phi_i)$, with parameters Θ and Φ_i is called the *new* model f_{new}. When a prediction for T_i is required the corresponding head model h_i is called. The distillation of the two models has the following objective:

$$f(x; \Theta, \Phi_c) = \begin{cases} f_{old}(x; \Theta, \Phi_i, h_i)[i], & 1 \leq i \leq n-1 \\ f_{new}(x; \Theta, \Phi_n, h_n)[i], & i = n \end{cases}$$

where i denotes the index of the considered task and f_{old} is the model trained on the previous tasks. The output of the consolidated model approximates the combination of the model outputs of the old model and the new model. To achieve this, the outputs of the old model and the new model are employed as supervisory signals in joint training of the consolidated model Φ_c.
To this purpose, they use the double distillation loss proposed by [37] to train a new adapter that is used with the pre-trained model to classify both old and newly learned tasks. The distillation process proceeds as follows: f_{old} and f_{new} are freezed, run a feed-forward pass for each training sample, and collect the *logits* of the two models $\hat{y}_{old} = \left[\hat{y}^1, \ldots, \hat{y}^{n-1}\right]$ and $\hat{y}_{new} = \hat{y}^n$ respectively, where the super-script is the class label associated with the neuron in the model. Then the difference between the logits produced by the consolidated model and the combination of logits generated by the two existing specialist models based on L_2-loss is minimized:

$$L_d(y, \hat{y}) = \frac{1}{n} \sum_{j=1}^{n} \left(y^j - \hat{y}^j\right)^2, \tag{1}$$

where y^j are the *logits* produced by the consolidated model for the j^{th} task and $\hat{y}$ is the concatenation of $\hat{y}_{old}$ and $\hat{y}_{new}$. The training objective for consolidation is given by

$$\min_{\Theta, \Phi_c} \frac{1}{|\mathcal{U}|} \sum_{x_j \in \mathcal{U}} L_d(y, \hat{y}), \tag{2}$$

where $\mathcal{U}$ denotes the training data used for distillation. After the consolidation, the adapter parameters Φ_c are used for f_{old} in the next round.

Transferability Estimation. ADA keeps a pool of adapters and the selection of the adapter to be distilled is based on transferability. The intuition behind this choice is that highly similar tasks will interfere less with each other

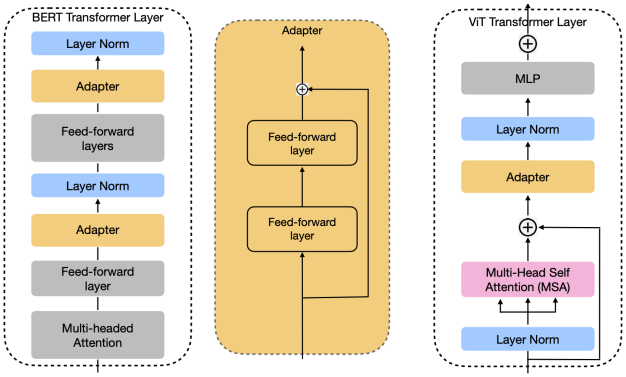

Figure 1. *Left* shows AdapterBERT [12] in a BERT transformer layer, and *Middle* shows the adapter architecture. Depending on configuration (Houlsby [12] or Pfeiffer [24]), only the top Adapter can be used. Right shows our Adapter implementation in a ViT transformer layer. As in AdapterBERT, we added an adapter before layer norm and feed-forward layers (MLP).

and so will cause significantly less forgetting. In [9] they leverage two common methods for transferability estimation: (1) *Log Expected Empirical Prediction* (LEEP) [22] and (2) TransRate [13].

LEEP is a measure (or a score) that can tell us, without training on the target data set, how effectively the transfer learning algorithms can transfer knowledge learned in the source model Θ_s to the target task, using the target data set $\mathcal{D}$. The details of LEEP can be checked in [22] and [9].

TransRate measures the transferability as the mutual information between the features of target examples extracted by a pre-trained model and labels of them with a single pass through the target data. It achieves minimal value when the data covariance matrices of all classes are the same. In this case, it is impossible to separate the data from different classes and no classifier can perform better than random guesses. To see the details how the knowledge transfer from a source task to a target task is measured, please see [13].

Algorithm. ADA procedure is detailed in Algorithm 1. For every new task the algorithm trains a new adapter and head model (called Φ_n and h_n). If the adapters pool did not reach the maximum size yet (controlled by K), it just adds it to the pool. If the pool reached the maximum size, the algorithm is forced to select one of the adapters already in the pool and distill it together with the newly trained one. In order to select which adapter to distill it leverages the transferability scores (e.g., LEEP or TransRate). Once the adapter in the pool with the highest transferability score (called $f_{old}^{j^*}$) is identified, it consolidates that adapter and the newly trained one into a new adapter and replaces the old one present in the pool. In order to be able to make effective predictions, the algorithm also keeps a mapping (in the map m) of which adapter in the pool must be used in combination with each of the task-specific heads.

Algorithm 1 Adaptive Distillation of Adapters (ADA)

Require: Θ: pre-trained model, K: adapters pool size
Freeze Θ
Create $m = Map()$
for $n \leftarrow 1$ to N **do**
  A task T_n is received
  Initialize Φ_n
  Process T_n and train new model $f_n(x; \Theta, \Phi_n)$ and head h_n
  Sample from T_n and add to distillation data $\mathcal{D}_{distill}$
  if $n \leq K$ **then**
    Set $f_n(x; \Theta, \Phi_1)$ to f_{old}^n
  else
    $j^* \leftarrow \arg\max_{j \in \{1,\dots,K\}}$ *TRANSCORE*(T_n, f_n, f_{old}^j)
    Add (n, j^*) to m
    Consolidate model:
      $f_{old}^{j^*}$ = *DISTILLATION*$(f_{old}^{j^*}, f_n, \mathcal{D}_{distill})$
  end if
  Serve predictions for any task $i \leq n$ using h_i and $f_{old}^{m(i)}$
end for

DISTILLATION$(f_{old}, f_n, \mathcal{D}_{distill})$:
  Get soft targets $\hat{y}_{old}$ from old model f_{old} with $\mathcal{D}_{distill}$
  Get soft targets $\hat{y}_{new}$ from new model f_n with $\mathcal{D}_{distill}$
  Initialize Φ_c
  Compute distillation loss as in Equation (1) and train model $f(x; \Theta, \Phi_c)$
  return f

4. Experiments

As discussed in Section 1, we would like to test ADA due to three main characteristics: i) limited or no data (external or synthetic data can be used for distillation) storage required; ii) almost-constant number of parameters added with the growing number of tasks and iii) interoperability with pre-trained Transformers available in public repositories. To this purpose, we designed a set of experiments on CIFAR100 [17] and MiniImageNet [28]. The complete setup is described in the next sections.

Experimental setup. Both CIFAR100 and MiniImageNet consist of 60000 colour images in 100 classes, with 600 images per class. We design two scenarios, in the first scenario each new task is a balanced binary classification problem where the positive class is selected at random and the data points in the negative class are selected randomly from the classes picked from the previous tasks. Each class can be selected to be the positive class only once. In the second scenario each task is a balanced multi-class classification problem with 5 classes. In both cases we provide the learner with 50 data points per class both at training and test time: in the first scenario each task will have a training set of 250 data points and in the second case of 100 data points. The total number of tasks is fixed to 20 for both scenarios. We use *Adam* as optimizer with the batch size of 8. For learning rate, we select best from $\{0.00005, 0.0001, 0.0005, 0.001\}$ after observing the results on the first five tasks.

Adapter architectures. We use pre-trained models from HuggingFace Transformers [32] as our base feature extractors. We ran experiments with ViT-B [6][1] and DeiT-B [29][2]. Both models use 12 layers of transformers block with a hidden size of 768 and number of self-attention heads as 12 and has around 86 M trainable parameters. We implemented the same adapter architecture for ViT and DeiT with AdapterBERT [12]. For all the adapter-based algorithms that we

[1] https://huggingface.co/google/vit-base-patch16-224

[2] https://dl.fbaipublicfiles.com/deit/deit_base_patch16_224-b5f2ef4d.pth

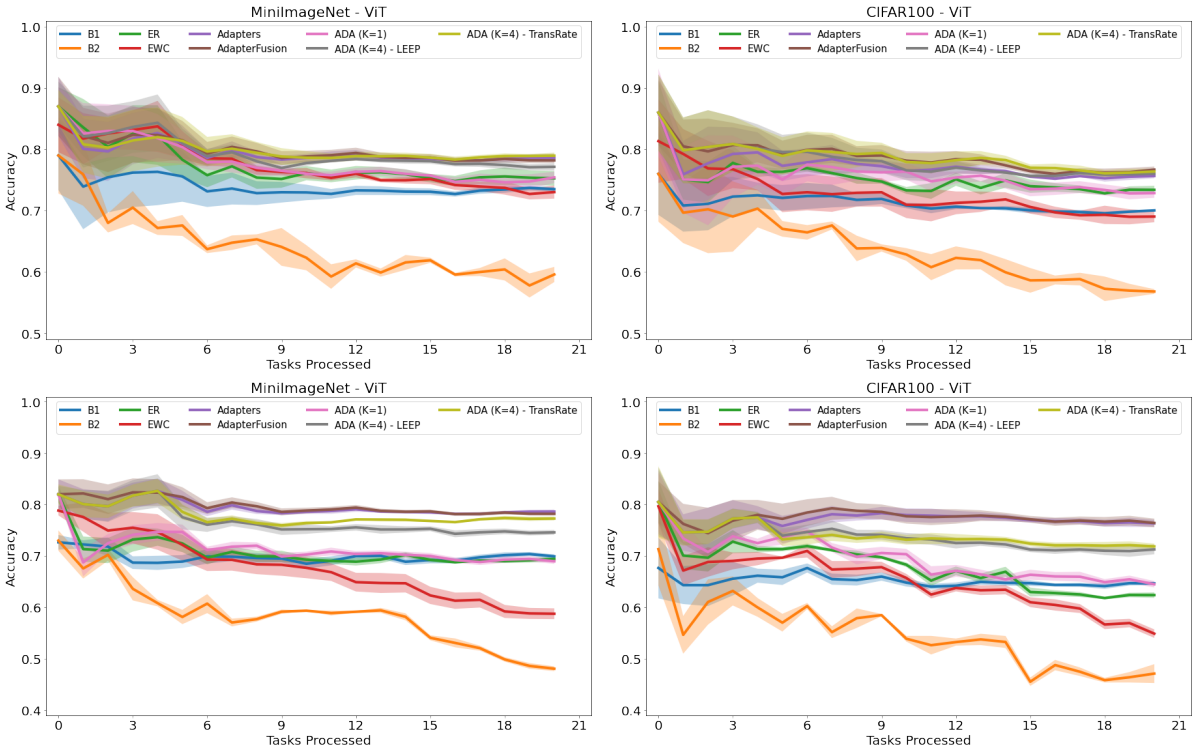

Figure 2. Comparison between baselines and ADA with ViT model on MiniImageNet and CIFAR100. Top figures shows the first scenario (binary) results, and bottom figures shows the second scenario (multi-class) results. On the x-axis we report the number of tasks processed, on the y-axis we report the average accuracy measured on the test set of the tasks processed, shaded area shows standard deviation.

define in the following section, we use the same configuration for the adapters, setting the adapter hidden size to 48. With this setting, an adapter contains 1.8 M parameters. We also train a head model for each task, that has the size of ViT-B$\times$ *output size*.

Baselines. We compare ADA with the following baselines. 1) *Fine-tuning head model (B1)*: We freeze the pre-trained representation and only fine-tune the output layer of each classification task. The output layer is multiple-head binary classifier that we also use for the other methods. 2) *Fine tuning the full model (B2)*: We fine-tune both the pre-trained representation and the output layer for each classification task. 3) *Adapters* [12]: We train and keep separate adapters for each classification task as well as the head models. 4) *AdapterFusion* [24]: It is a two stage learning algorithm that leverages knowledge from multiple tasks by combining the representations from several task adapters in order to improve the performance on the target task. This follows exactly the solution depicted in Section 3, Adapters. 5) *Experience Replay (ER)* [27]: ER is a commonly used baseline in Continual Learning that stores a subset of data for each task and then "replays" the old data together with the new one to avoid forgetting old concepts. To make the results comparable, we use ER with same pre-trained network of the other algorithm and a single adapter being trained. Memory size is set to 500. 6) *Elastic Weight Consolidation (EWC)* [16]: EWC is a regularization-based CL method that assumes that some weights of the trained neural network are more important for previously learned tasks than others. During training of the neural network on a new task, changes to the weights of the network are made less likely the greater their importance. To estimate the importance of the network weights, EWC uses probabilistic mechanisms, in particular the Fisher information matrix. We tune the regularization coefficient of EWC by grid search in $\{0, 1, 10, 100, 1000\}$.
In addition to these baselines, we use one special case of ADA with K=1, offering another reference point with the performance of ER and helping to quantify the advantage of effective consolidation of adapters. All the results in this section are average of 5 runs.

Predictive performance. Figure 2 shows the comparison of ADA and the baseline methods. It can be clearly seen that freezing all pre-trained model parameters, and fine-tuning only the head models (B1) leaded to an inferior performance compared to other approaches. The main reason is that the

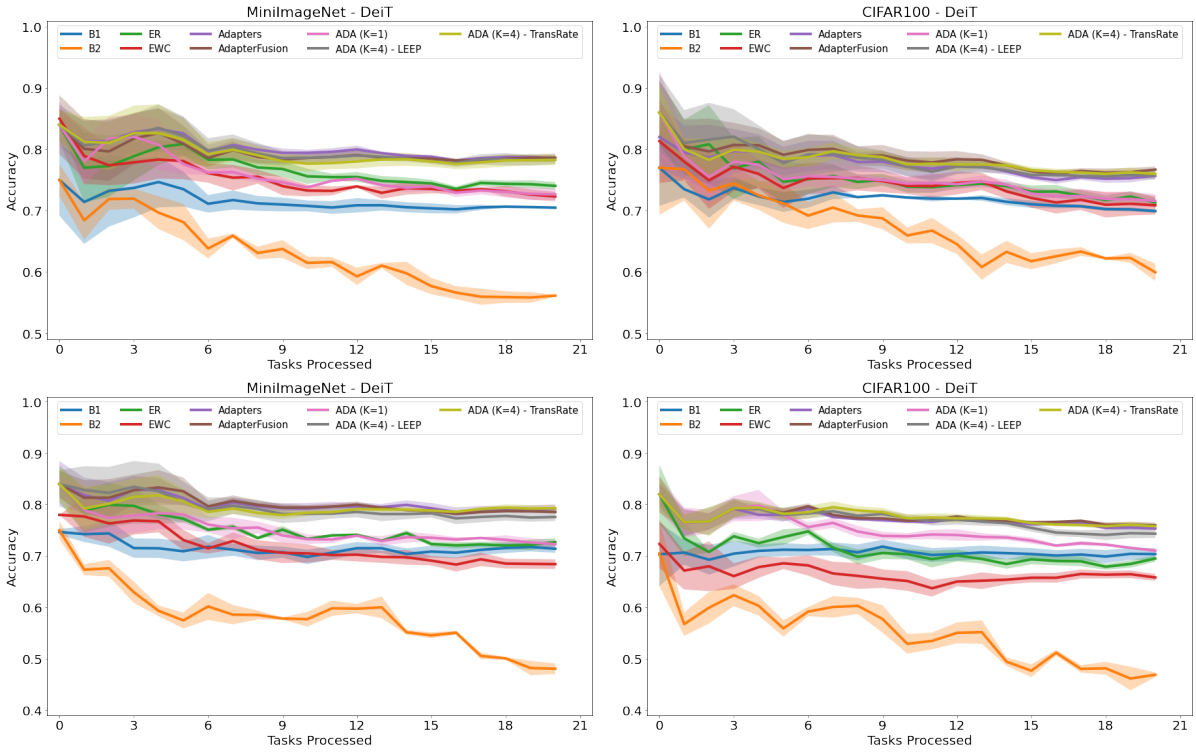

Figure 3. Comparison between baselines and ADA with DeiT model on MiniImageNet and CIFAR100. Top figures shows the first scenario (binary) results, and bottom figures shows the second scenario (multi-class) results.

head models have small amount of parameters to train and fine-tuning only the heads suffers from under-fitting. B2 performs well only for first 2-3 tasks, since we keep training the complete model, it forgets the previously learned tasks very quickly. Adapters and AdapterFusion add 1.8 M parameters for each task and train these parameters with new task data, fixing them after training. There is no forgetting since the parameters are not shared among tasks and the results on each dataset confirm this. Although the careful tuning of regularization coefficient, EWC cannot handle CF, especially for multi-class classification problem. ADA with K=1 shows that distillation alone doesn't prevent forgetting. In almost all cases, ER perform on-par with ADA K=1, providing evidence that a small amount of memory can actually improve performance compared to fine-tuning or regularization, but the improvement is limited and does not last as the number of tasks increases.

ADA-LEEP and ADA-TransRate results with K=4 adapters show that selective consolidation of adapters significantly improve the performance. For binary classification, their performance are on par with AdapterFusion while the number of model parameters is significantly lower. For multi-class, their performance slightly declines after a certain number of tasks. This is discussed in [9], and the main reason is that the capacity of the adapter is exceeded. Using a bigger adapter pool, or using larger adapters can solve the issue quickly, but to keep the comparison fair, we used the same size adapters for each algorithm.

To validate the interoperability of ADA to different models, we run the same experiments on DeiT model. Figure 3 demonstrates the same behaviour of algorithms with DeiT.

	Fine-Tuning (B1, B2) and EWC			
	Trainable	Inference	Total	Total (Size)
Task = $\{1, 10, 20\}$	86 M	86 M	86 M	344

	Adapters & AdapterFusion			
	Trainable	Inference	Total	Total (Size)
Task = 1	1.8 M	87.8 M	87.8 M	351.2
Task = 10	1.8 M	86 + (F×1.8) M	104 M	416
Task = 20	1.8 M	86 + (F×1.8) M	122 M	488

	ADA			
	Trainable	Inference	Total	Total (Size)
Task = 1	1.8 M	87.8 M	87.8 M	351.2
Task = 10	2×1.8 M	87.8 M	86 + (K+1)×1.8 M	344 + (K+1)×7.2
Task = 20	2×1.8 M	87.8 M	86 + (K+1)×1.8 M	344 + (K+1)×7.2

Table 1. The number of all parameters and those used for training and inference as well as the model size of methods for ViT-B (Same for Deit-B). K is the number of adapters in the pool, and F is the number of fused adapters (it is between 2 and number of tasks). For the *Adapters* $F = 1$. For ER, for Task = $\{1, 10, 20\}$, it is same with ADA Task=1. Total size is in MB.

Memory consumption. Table 1 reports the number of parameters used for baselines and ADA in our experiments. We don't add the head size to the table, since it's very small: 768 parameters per binary head, 15K parameters (6 KB) for 20 tasks, 3840 per multi-class head, 75K parameters (30KB) for 20 tasks. Also they are same for all the methods.
These results make clear that ADA is significantly more efficient in terms of memory usage. It can achieve predictive performance similar to the one of Adapters and AdapterFusion while requiring significantly less model parameters. ADA stores only 5 Adapters (K=4 adapters in the pool, and one adapter for new task), against the 20 required by AdapterFusion.

Similarities and differences with the NLP experiments In [9], they address incremental binary text classification. In this paper, we further investigated the multi-class classification where we observe a total of 100 classes. Almost all algorithms show similar behaviour with NLP experiments. One interesting outcome is the performance of full fine-tuning (B2) performed better with image transformers. The accuracy declines more significantly on NLP datasets with NLP transformers. Besides LEEP works better with image datasets as it was originally proposed for CV tasks. There is mostly no difference between ADA-LEEP and ADA-TransRate, or the difference is very small. But in [9], the difference is noticable.

5. Conclusion

In this work we show that the usage of Adapters in combination with Transformers for continual CV problems. In particular, utilizing ad-hoc algorithms, such as ADA, can give a strong result in terms of predictive performance with constant parameter increase. Vision transformers have been known to have a tendency to overfit training datasets, consequently leading to poor predictive performance in small data regimes, however [23] just showed that this claim is poorly supported, explains the nature of multi-head self-attentions and shows that ViT does not overfit even on smaller datasets. This work is encouraging for the future studies in CV with vision transformers, and any development in that field will positively impact CL research with Transformers.

There are some aspects of our results which we would like to further investigate in the future. For instance, in this work we adopted the same Adapters structure leverage in the NLP domain and, while that gave good results, there is the possibility that it is a suboptimal choice. Also, we tested ADAwith a fixed number of Adapters but it is easy to observe that the number of Adapters could be controlled by the algorithm itself, for example leveraging heuristics based on thresholding the LEEP or TransRate scores.

References

[1] Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. *arXiv preprint arXiv:1806.05421*, 2018. 1

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[3] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and M Ranzato. Continual learning with tiny episodic memories. 2019. 1

[4] Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. *arXiv preprint arXiv:1906.01076*, 2019. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 3, 4

[7] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pages 86–102. Springer, 2020. 1

[8] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. *arXiv preprint arXiv:2111.11326*, 2021. 2

[9] Beyza Ermis, Giovanni Zappella, Martin Wistuba, and Cedric Archambeau. Memory efficient continual learning for neural text classification. *arXiv preprint arXiv:2203.04640*, 2022. 2, 4, 6, 7

[10] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019. 1

[11] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019. 1

[12] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019. 2, 3, 4, 5

[13] Long-Kai Huang, Ying Wei, Yu Rong, Qiang Yang, and Junzhou Huang. Frustratingly easy transferability estimation. *arXiv preprint arXiv:2106.09362*, 2021. 4

[14] Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. Continual learning for text classification with information disentanglement based regularization. *arXiv*

preprint arXiv:2104.05489, 2021. 1

[15] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 1

[16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 1, 2, 5

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

[18] Duo Li, Guimei Cao, Yunlu Xu, Zhanzhan Cheng, and Yi Niu. Technical report for iccv 2021 challenge sslad-track3b: Transformers are better continual learners. *arXiv preprint arXiv:2201.04924*, 2022. 1, 2

[19] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934. PMLR, 2019. 1

[20] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017. 1

[21] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[22] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *International Conference on Machine Learning*, pages 7294–7305. PMLR, 2020. 4

[23] Namuk Park and Songkuk Kim. How do vision transformers work? *arXiv preprint arXiv:2202.06709*, 2022. 7

[24] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online, Apr. 2021. Association for Computational Linguistics. 2, 3, 5

[25] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018. 2

[26] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 1

[27] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018. 1, 2, 5

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 4

[29] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 3, 4

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[31] Zirui Wang, Sanket Vaibhav Mehta, Barnabás Póczos, and Jaime Carbonell. Efficient meta lifelong-learning with limited memory. *arXiv preprint arXiv:2010.02500*, 2020. 1

[32] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020. 4

[33] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. 1

[34] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021. 1

[35] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. *arXiv preprint arXiv:1909.00161*, 2019. 1

[36] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017. 1

[37] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140, 2020. 3