CLiMB: A Continual Learning Benchmark for Vision-and-Language Tasks

Tejas Srinivasan[1] **Ting-Yun Chang**[1] **Leticia Pinto Alva**[1]
Georgios Chochlakis[1] **Mohammad Rostami**[1,2] **Jesse Thomason**[1]
[1]University of Southern California [2]USC Information Sciences Institute
`{tejas.srinivasan,tingyun,pintoalv,chochlak,rostamim,jessetho}@usc.edu`

Abstract

Current state-of-the-art vision-and-language models are evaluated on tasks either individually or in a multi-task setting, overlooking the challenges of continually learning (CL) tasks as they arrive. Existing CL benchmarks have facilitated research on task adaptation and mitigating "catastrophic forgetting," but are limited to vision-only and language-only tasks. We present CLiMB, a benchmark to study the challenge of learning multimodal tasks in a CL setting, and to systematically evaluate how upstream continual learning can rapidly generalize to new multimodal and unimodal tasks. CLiMB includes implementations of several CL algorithms and a modified Vision-Language Transformer (ViLT) model that can be deployed on both multimodal and unimodal tasks. We find that common CL methods can help mitigate forgetting during multimodal task learning, but do not enable cross-task knowledge transfer. We envision that CLiMB will facilitate research on a new class of CL algorithms for this challenging multimodal setting.

1 Introduction

Large-scale pre-trained models, including crossmodal vision-and-language models, are generally fine-tuned on each downstream task individually, requiring fine-tuning and storing new models for each task. By contrast, multi-task learning requires fixing a set of tasks, but such training is incapable of dynamically learning new tasks. Although continual learning (CL) algorithms have explored cross-task knowledge transfer, existing methods primarily consider unimodal tasks in artificial settings [Jin et al., 2021, Lin et al., 2021]. Multimodal pre-training can encode useful and transferable features for diverse tasks, but learning from a *sequence* of different multimodal tasks and the subsequent forgetting effects [Kirkpatrick et al., 2017] have not yet been studied.

Additionally, it is assumed that these deployed models will encounter tasks containing all modalities seen during training time. This assumption means learning separate models for language-only, vision-only, and vision-language tasks, as opposed to a single "generalist" model that can handle all modalities or subsets of them [Reed et al., 2022]. Yet, existing work suggests that knowledge grounded in multiple modalities can benefit unimodal tasks [Desai and Johnson, 2021, Jin et al., 2022]. Currently, the research community lacks a suitable benchmark to systematically study how models can continually learn vision-and-language tasks while being transferable to unimodal tasks.

In this paper, we introduce the **Continual Learning in Multimodality Benchmark (CLiMB)**,[1] to facilitate the study of CL in vision-and-language tasks with deployment to multi- and unimodal tasks.

[1]The code for our benchmark is available at `https://github.com/GLAMOR-USC/CLiMB`

Submitted to the 36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks. Do not distribute.

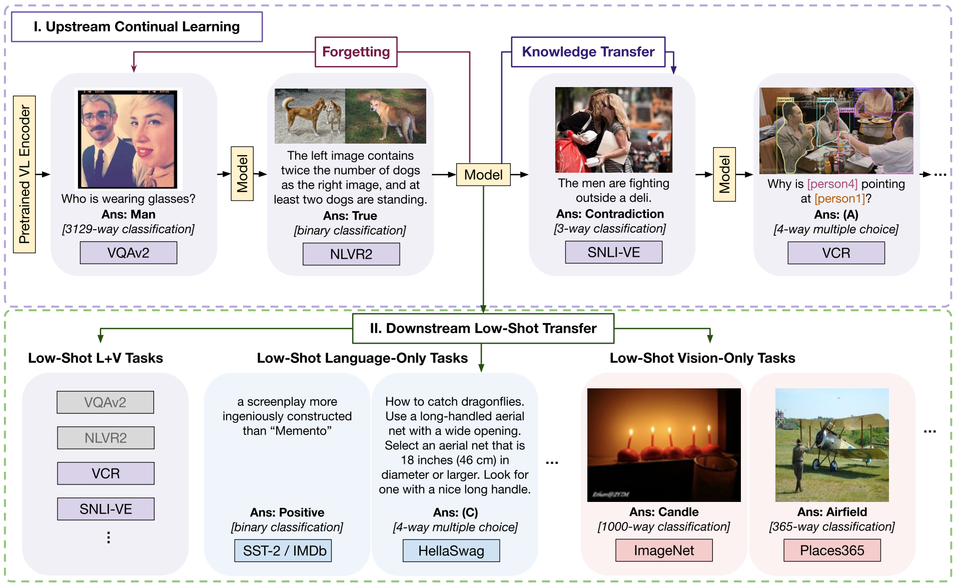

I. Upstream Continual Learning
Forgetting
Knowledge Transfer
Pretrained VL Encoder
Who is wearing glasses?
Ans: Man
[3129-way classification]
VQAv2
Model
The left image contains twice the number of dogs as the right image, and at least two dogs are standing.
Ans: True
[binary classification]
NLVR2
Model
The men are fighting outside a deli.
Ans: Contradiction
[3-way classification]
SNLI-VE
Model
Why is [person4] pointing at [person1]?
Ans: (A)
[4-way multiple choice]
VCR
II. Downstream Low-Shot Transfer
Low-Shot L+V Tasks
VQAv2
NLVR2
VCR
SNLI-VE
Low-Shot Language-Only Tasks
a screenplay more ingeniously constructed than "Memento"
Ans: Positive
[binary classification]
SST-2 / IMDb
How to catch dragonflies. Use a long-handled aerial net with a wide opening. Select an aerial net that is 18 inches (46 cm) in diameter or larger. Look for one with a nice long handle.
Ans: (C)
[4-way multiple choice]
HellaSwag
Low-Shot Vision-Only Tasks
Ans: Candle
[1000-way classification]
ImageNet
Ans: Airfield
[365-way classification]
Places365

Figure 1: CLiMB evaluates candidate CL models and learning algorithms in two phases. For Phase I, Upstream Continual Learning, a pre-trained multimodal model is trained on a sequence of vision-and-language tasks, and evaluated after each task on its degree of Forgetting of past tasks and Knowledge Transfer to the next task. For Phase II, after each multimodal task the model is evaluated for its Downstream Low-Shot Transfer capability on both multimodal and unimodal tasks.

We formulate a learning problem wherein a model is first trained on sequentially arriving vision-and-language tasks, referred to as **upstream continual learning**, and then **transferred downstream to low-shot** multimodal and unimodal tasks. CLiMB initially includes four vision-and-language tasks, five language tasks, and four vision tasks, and is extensible to new tasks, models, and learning algorithms. Experiments using CLiMB find that existing CL algorithms can mitigate forgetting, but not transfer knowledge across tasks, revealing a need for new research into continual learning strategies for vision-language tasks. Further, current CL algorithms and multimodal models are not well suited for low-shot adaptation to multimodal or unimodal tasks. We hope CLiMB will provide the basis for developing models and learning algorithms for multimodal continual learning.

2 Background and Related Work

Continual, or lifelong, learning is an essential ability to develop autonomous agents that can learn in a cumulative way [Chen and Liu, 2018]. In CL, a model is trained on sequentially arriving tasks and evaluated both on its ability to learn future tasks as well as retain performance on past learned tasks [Kirkpatrick et al., 2017]. A necessity for developing CL algorithms is benchmarks that collate suitable sequential tasks. There are two primary approaches to create such CL benchmarks.

The first approach is to split existing tasks into non-overlapping sub-tasks that are sequentially presented for continual learning. For example, one can divide tasks along input categories [Greco et al., 2019] or output classes [Vinyals et al., 2016, Kirkpatrick et al., 2017] into disjoint sets. Mimicking real world distribution shift, timestamps can group data instances according to the order of their creation [Lin et al., 2021].

CLiMB goes beyond such artificial, single-task-based CL and instead aggregates several diverse tasks. Similarly, unimodal benchmarks such as Visual Domain Decathlon [Rebuffi et al., 2017] and Natural Language Decathlon [McCann et al., 2018] aggregate 10 image classification and 10 language tasks, respectively. The CLIF-26 benchmark [Jin et al., 2021] is built for CL on the GLUE [Wang et al., 2019] language tasks. CLiMB goes beyond these unimodal benchmarks by evaluating on sequences of multimodal, vision-and-language tasks *and* testing downstream transfer to unimodal tasks.

Task	**Vision Input(s)**	**Language Input(s)**	**Decision**	**Score Metric**
VQAv2	Image	Question	1 of 3129	VQAScore[2]
NLVR2	2 images	Caption	True/False	Accuracy
SNLI-VE	Image	Hypothesis	Ent/Neu/Con	Accuracy
VCR	Image w/ bboxes	Question + 4 Answers	1 of 4	Accuracy
IMDb		Sentence	Pos/Neg	Accuracy
SST-2		Sentence	Pos/Neg	Accuracy
HellaSwag		Sentence Prefix + 4 Endings	1 of 4	Accuracy
CommonsenseQA		Question + 5 Answers	1 of 5	Accuracy
PIQA		Question + 2 Answers	1 of 2	Accuracy
ImageNet-1000	Image		1 of 1000	Top-1 Accuracy
iNaturalist2019	Image		1 of 1010	Top-1 Accuracy
Places365	Image		1 of 365	Top-1 Accuracy
COCO-object	Image		n of 80	Micro-F1

Table 1: The initial tasks in CLiMB include various forms of vision and language inputs, and each task is framed as a classification problem. Multimodal vision-and-language tasks serve as upstream training for both multimodal and unimodal downstream, low-shot tasks.

3 CLiMB: The Continual Learning in Multimodality Benchmark

CLiMB tests the ability of models and learning algorithms to adapt to a sequentially arriving stream of vision-language tasks, as well as rapidly transfer to new multimodal and unimodal tasks (Table 1).

3.1 CLiMB Learning and Evaluation

Learning and evaluation in CLiMB proceeds in two phases: **upstream continual learning** and **downstream low-shot transfer** (Figure 1). Table 2 summarizes our CL evaluation metrics. We denote a task with modality $M \in \{V, L, VL\}$ as $\mathcal{T}_M^i$ and the number of such tasks as K_M.

Upstream Continual Learning of Multimodal Tasks A candidate model $\mathcal{M}$ encounters a sequence of K_{VL} vision-language tasks, $\mathcal{T}_{VL}^{1...K_{VL}}$. $\mathcal{M}$ can be initialized with a pre-trained encoder. We allow parameter additions to the base model on a per-task basis. In this work we add task-specific classification layers for each new task on top of the base encoder model. The model $\mathcal{M}$ is sequentially trained on the training split of each task $\mathcal{T}_{VL}^i$ with candidate CL algorithm $\mathcal{A}$. For task $\mathcal{T}_{VL}^i$, the model is not presented with any inputs from $\mathcal{T}_{VL}^{1...i-1}$. However, the CL algorithm $\mathcal{A}$ may allocate memory to access previous training examples.

We evaluate two primary model properties in the upstream phase: **upstream knowledge transfer** from past learned tasks to new tasks, and withstanding **forgetting** of previously-seen tasks. The upstream knowledge transfer $\mathbb{T}_{UK}(i)$ on task $\mathcal{T}_{VL}^i$ is the relative gain in score from learning the previous tasks $\mathcal{T}_{VL}^{1...i-1}$. Forgetting $\mathbb{T}_F(j \leftarrow i)$ of previously-seen task $\mathcal{T}_{VL}^j$ is the relative performance degradation in that task after learning subsequent tasks $\mathcal{T}_{VL}^{j+1...i}$ (Table 2).

Downstream Transfer to Low-Shot Tasks We evaluate the low-shot adaptation ability of the model $\mathcal{M}$ after learning each upstream vision-language task. After training on the i^{th} upstream task $\mathcal{T}_{VL}^i$, we evaluate low-shot transfer to remaining multimodal tasks $\mathcal{T}_{VL}^{i+1...K_{VL}}$, as well as unimodal tasks $\mathcal{T}_V^{1...K_V}$ and $\mathcal{T}_L^{1...K_L}$. Specifically, for every task in each modality, a low-shot instance of task $\mathcal{T}_M^i$, denoted as $\mathcal{T}_M^{LS(i)}$, is created. The **low-shot transfer** ability to this task is evaluated by fine-tuning upstream encoder checkpoints on task $\mathcal{T}_M^{LS(i)}$. We compute the low-shot transfer $\mathbb{T}_{LS}^M(i)$ as the relative improvement of the CL encoder's performance on the low-shot task $\mathcal{T}_M^{LS(i)}$, denoted as $S_{\mathcal{A}}^{LS(i)}$, over the pre-trained encoder's performance on the same low-shot task, $S_{PT}^{LS(i)}$.

[2]https://visualqa.org/evaluation.html

Evaluation Type	Description	Metric ($\times 100\%$)
Upstream Knowledge Transfer, $\mathbb{T}_{UK}(i)$	Improvement of performance on task $\mathcal{T}_{VL}^{i}$ after training on tasks $\mathcal{T}_{VL}^{1\dots i}$ using algorithm $\mathcal{A}$ ($S_{\mathcal{A}}^{i}$) compared to finetuning the pretrained model on $\mathcal{T}_{VL}^{i}$ directly (S_{PT}^{i}).	$\mathbb{T}_{UK}(i) = \frac{S_{\mathcal{A}}^{i} - S_{PT}^{i}}{S_{PT}^{i} - S_{R}^{i}}$
Forgetting Transfer, $\mathbb{T}_{F}(j \leftarrow i)$	Decrease of performance when a model trained on tasks $\mathcal{T}_{VL}^{1\dots i}$ is evaluated on task $\mathcal{T}_{VL}^{j}(j < i)$. $S_{\mathcal{A}}^{j \leftarrow i}$ denotes model performance on $\mathcal{T}_{VL}^{j}$ after training up to i.	$\mathbb{T}_{F}(j \leftarrow i) = \frac{S_{\mathcal{A}}^{j} - S_{\mathcal{A}}^{j \leftarrow i}}{S_{\mathcal{A}}^{j} - S_{R}^{j}}$
Low-Shot Transfer, $\mathbb{T}_{LS}^{M}(i)$	Improvement of performance on low-shot task $\mathcal{T}_{M}^{LS(i)}$ using an encoder checkpoint trained by upstream algorithm $\mathcal{A}$ ($S_{\mathcal{A}}^{LS(i)}$) compared to learning the same low-shot task without any upstream learning ($S_{PT}^{LS(i)}$).	$\mathbb{T}_{LS}^{M}(i) = \frac{S_{\mathcal{A}}^{LS(i)} - S_{PT}^{LS(i)}}{S_{PT}^{LS(i)} - S_{R}^{i}}$

Table 2: Model and algorithm evaluation metrics in the upstream and downstream phases. For the i^{th} task, we compute a model's $\delta^i = S^i - S_R^i$, the model's task score S^i minus the score S_R^i of random selection. Our evaluation protocol computes each metric as a relative change in the δ^i of a CL algorithm $\mathcal{A}$ over a baseline setting $\mathcal{B}$ to enable across-task comparisons. Each evaluation metric is calculated as $\frac{\delta_{\mathcal{A}} - \delta_{\mathcal{B}}}{\delta_{\mathcal{B}}} = \frac{S_{\mathcal{A}} - S_{\mathcal{B}}}{S_{\mathcal{B}} - S_R}$, and is presented as a percentage.

3.2 CLiMB Multimodal and Unimodal Tasks

CLiMB initially includes four multimodal upstream vision-language tasks, five language-only tasks, and four vision-only tasks (Table 1). We frame each as a classification task.

Vision-Language Tasks CLiMB includes VQAv2 [Goyal et al., 2017], NLVR2 [Suhr et al., 2019], SNLI-VE [Xie et al., 2019] and VCR [Zellers et al., 2019a]. Solving these challenging tasks requires different kinds of knowledge in the multimodal model: question answering, visual and commonsense reasoning, entailment understanding.

Language-Only Tasks CLiMB includes IMDb [Maas et al., 2011], SST-2 [Socher et al., 2013], HellaSwag [Zellers et al., 2019b], CommonsenseQA [Talmor et al., 2019], and PIQA [Bisk et al., 2020]. We hypothesize that visually-grounded knowledge from upstream tasks may benefit tasks such as IMDb and SST-2, which are sourced from movie reviews, as well as HellaSwag, sourced from video captions, and PIQA, sourced from physically-grounded instructions from images and videos. Further, commonsense knowledge and reasoning skills obtained from VCR and NLVR2 may benefit tasks like HellaSwag, CommonsenseQA, and PIQA.

Vision-Only Tasks CLiMB includes ImageNet-1000 [Russakovsky et al., 2015], iNaturalist2019 [Van Horn et al., 2018], Places365 [Mahajan et al., 2018], and MS-COCO object detection (formulated as a multi-label classification task). Since VQAv2 images are sourced from MS-COCO [Lin et al., 2014], we hypothesize VQAv2 upstream learning may aid in the COCO object detection task.

4 Modeling and Experiments

Using CLiMB, we study the performance of several commonly used CL algorithms on multimodal tasks. We use fixed upstream task order (VQAv2 $\rightarrow$ NLVR2 $\rightarrow$ SNLI-VE $\rightarrow$ VCR).

4.1 Vision-Language Encoder: ViLT

We use a pre-trained Vision-Language Transformer (ViLT) [Kim et al., 2021] as a backbone encoder. Unlike other pre-trained vision-language models [Lu et al., 2019, Chen et al., 2020] that build upon region-level features extracted from Faster R-CNN [Ren et al., 2015], ViLT directly operates on image patches without using any convolutional layers. In ViLT, text tokens and image patches are concatenated into an input sequence and passed through a Transformer, which learns the vision-language alignment with self-attention across both modalities (Figure 2).

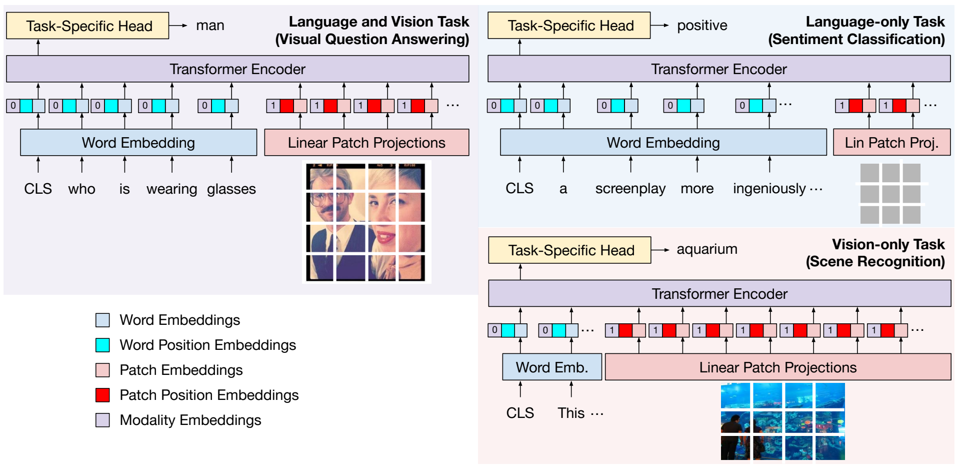


Figure 2: The ViLT model [Kim et al., 2021] operates on vision and language inputs (left). We adapt these inputs for language-only tasks by providing the average MS-COCO image as in-domain visual input, and vision-only tasks by providing vacuous language input "This is an image" (right).

4.2 Upstream Experiments: Algorithms and Task Ordering

CLiMB includes several CL algorithm implementations. **Sequential Fine-tuning (SeqFT)** fine-tunes the full encoder and task-specific layers for each task in order. This baseline algorithm is an extension of the single-task fine-tuning paradigm to the CL setting. We also experiment with a **Frozen Encoder** baseline that trains only the task-specific layers. Fine-tuning all parameters may cause forgetting since the encoder parameters are overwritten, while fine-tuning only the task-specific layer prevents knowledge transfer since the shared encoder parameters are fixed. In **Frozen Bottom-K**, we freeze the bottom K layers of the encoder and fine-tune the rest, balancing these solutions (we set K=9).

CLiMB also includes two CL algorithms that fine-tune all parameters but are designed to mitigate forgetting. **Experience Replay (ER)** [Chaudhry et al., 2019] caches a small percentage of training examples in a memory buffer after training on each task, and periodically performs a "replay" training step using cached examples to refresh the model. **Elastic Weight Consolidation (EWC)** [Kirkpatrick et al., 2017] is a regularization method that adds an auxiliary L2 loss between weights in the current model and previous checkpoints to slow change in important encoder parameters.

Finally, CLiMB includes **Adapters** [Houlsby et al., 2019], which add a small number of task-specific parameters, called Adapter modules, within layers of the pre-trained encoder. During training, the encoder's original parameters are kept frozen and only the Adapter modules are trained. We use a new set of Adapter modules each time we train on a new task, which leaves the previous tasks' modules untouched and prevents forgetting, but also does not facilitate cross-task knowledge transfer.

4.3 Downstream Low-Shot Experiments

We consider low-shot multimodal and unimodal tasks. We first define low-shot settings for different task types, then explain how we apply the multimodal ViLT model to unimodal settings.

Low-Shot Task Settings We study "low-shot" training paradigms where only a fraction of full training data is available. For the multimodal classification tasks, NLVR2 and SNLI-VE, we use 2048 examples per class, whereas for the multiple choice VCR task, we use 5% of the training data. Among unimodal tasks, for vanilla classification tasks (IMDb, SST2, ImageNet-1000, iNaturalist, and Places365), we consider training with $N = \{16, 32\}$ examples per class. For multiple choice classification tasks (PIQA, CommonsenseQA, HellaSwag), we use $N = \{1024, 4096\}$ since these tasks are considerably more challenging. For the multi-label COCO object detection task, we consider $M = \{5\%, 10\%\}$ of the original training data.

Alg $\mathcal{A}$	Params Trained	Task 1 VQAv2	Task 2 NLVR2	Task 3 SNLI-VE	Task 4 VCR
Direct FT	100%	[67.70]	[73.07]	[76.31]	[61.31]
SeqFT	100%	0.13% [67.79]	-1.80% [72.66]	-3.33% [74.89]	-5.09% [59.47]
Frozen Enc	7.88%	-14.10% [58.15]	-40.78% [63.66]	-15.98% [69.45]	-53.47% [41.90]
Frozen B9	25.92%	-0.58% [67.30]	-0.58% [72.94]	-3.31% [74.90]	-15.49% [55.69]
ER	100%	0.26% [67.87]	0.56% [73.20]	-2.89% [75.08]	-4.45% [59.70]
EWC	100%	0.20% [67.84]	-2.79% [72.39]	-4.52% [74.38]	-4.86% [59.55]
Adapters	13.02%	**0.59% [68.10]**	**2.55% [73.66]**	**-0.56% [76.08]**	**-0.36% [61.18]**

Table 3: Upstream Knowledge Transfer $\mathbb{T}_{UK}(i)$ relative to direct fine-tuning on each task, along with task score $[S^i_{\mathcal{A}}]$ (%), for different CL algorithms $\mathcal{A}$ applied to ViLT. No CL algorithms achieve notable positive Knowledge Transfer, while the majority in fact *hurt* learning of new tasks.

Unimodal Tasks in ViLT To apply ViLT to vision-only tasks, we use the phrase "This is an image" as the language input paired with the input image from the vision task. For language-only tasks, however, we need to address several challenges to effectively apply ViLT.

First, we find that averaging all MS-COCO training images into a single, in-distribution image paired with text inputs produces better results with ViLT than not concatenating any image tokens at all to the Transformer input sequence.

Second, ViLT only allows a maximum of 40 language tokens in the input, which is enough for captions but insufficient for most language tasks. To deal with this challenge, we first downsample the vacuous image to reduce its token length from 144 to 16. Next, we extend the available language tokens by creating copies of pre-trained ViLT's language positional embeddings, $E \in \mathbb{R}^{40\times d}$, and concatenating these copies to get extended positional embeddings, $\hat{E} \in \mathbb{R}^{L\times d}$, where L is the maximum sequence length of each task and d is the embedding dimension.

Finally, ViLT's language pre-training is on image captions that do not represent more general language use. We additionally experiment with a ViLT-BERT [Chochlakis et al., 2022] model that extracts language representations from a pre-trained, frozen BERT [Devlin et al., 2019] that serve as input embeddings for ViLT. Please refer to the supplementary materials for more experiments and details.

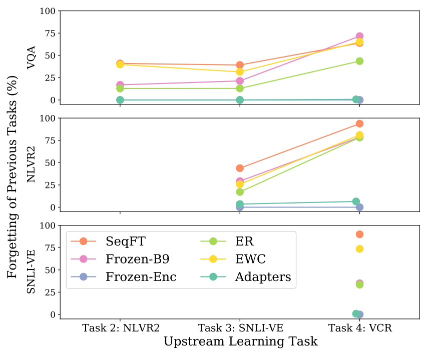
5 Results

We present Knowledge Transfer and Forgetting capabilities of different CL algorithms, experiments with multiple upstream task orders, and Low-Shot Transfer to downstream tasks.

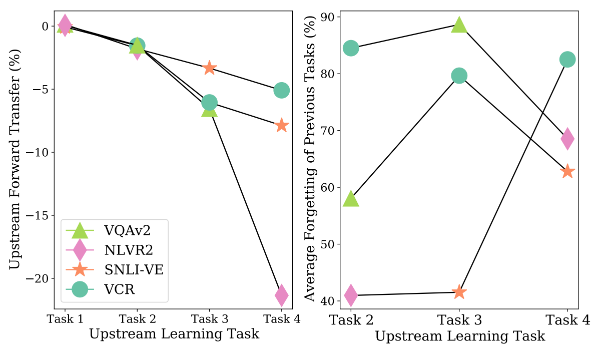
5.1 Upstream Learning Results

We find that common CL algorithms do not facilitate positive knowledge transfer in the vision-and-language setting of CLiMB, and in fact often hurt future task learning. Some are able to effectively mitigate forgetting, but none perform as well as directly fine-tuning on a candidate task. By examining the effects of task order, we conclude that the VCR task hurts further upstream task learning.

Upstream Knowledge Transfer In Table 3, we compare the upstream knowledge transfer exhibited by the different algorithms described in Section 4.2. Freezing the entire encoder severely underperforms the direct fine-tuning baseline for each task. Among other methods, all perform similarly to directly fine-tuning on the first task, with approximately zero knowledge transfer. However, for all methods other than Adapters, more continual learning hurts the model's ability to learn new tasks, as evidenced by the increasingly negative upstream transfer for later tasks. This effect may be due to loss of pre-training knowledge which is useful for task adaptation. Adapters, which do not train shared encoder parameters, do not exhibit this negative knowledge transfer, and show comparable performance to full model fine-tuning despite having very few learnable parameters.



(a)



(b)

Figure 3: **(a)** Forgetting $\mathbb{T}_F(j \leftarrow i)$ (%) of the previous $i-1$ tasks for each algorithm. Each subplot denotes model performance on one of the previous tasks. While all algorithms that fine-tune shared parameters exhibit Forgetting, ER best preserves past task performance. **(b)** Effect of task order on upstream Knowledge Transfer (left) and Forgetting (right) for three different orders. Lines represent performance conditioned on a particular task order. After experiencing the VCR task, models exhibit lower Knowledge Transfer and higher Forgetting.

Forgetting Figure 3a shows how each algorithm affects forgetting of previous tasks. Sequential Fine-tuning forgets previous tasks to a large extent, Frozen Bottom-9 shows slight improvement, and freezing the encoder prevents forgetting entirely. Experience Replay does a better job at retaining task performance, while EWC shows only a slight improvement. Adapters enable a model to learn upstream tasks in the multimodal CL setting *while not forgetting tasks already learned*, adding only 3-4% parameters per task. Interestingly, forgetting is more severe after training models on the VCR task, demonstrating that the order of encountering tasks affects continual learning.

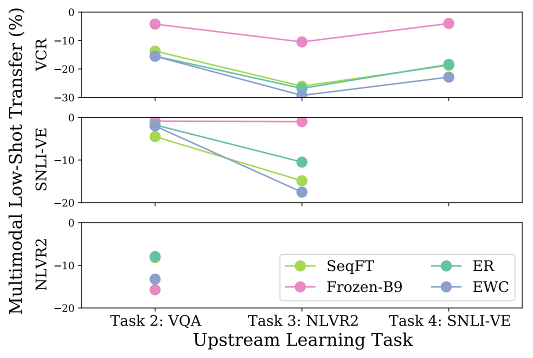


Figure 4: Low-shot transfer, $\mathbb{T}_{LS}^{VL}(j)$, for multimodal tasks $j = \{i+1, ..., K_{VL}\}$ after training on upstream task $\mathcal{T}_{VL}^{i}$. All CL algorithms exhibit negative Low-shot transfer on all multimodal tasks.

Effect of Upstream Task Order Figure 3b shows the upstream knowledge transfer and forgetting for ViLT using Sequential Fine-tuning on three different task orders. While the upstream transfer is similar for the first two tasks in each task ordering, training on VCR negatively affects both knowledge transfer to future tasks and forgetting of past tasks. This effect may be due to a shift in the visual domain of VCR: input images have colored boxes drawn on them to represent grounded objects in the question, following previous work [Zellers et al., 2021, Hessel et al., 2022].

5.2 Downstream Low-Shot Transfer Results

In downstream transfer, we fine-tune the entire model irrespective of the upstream CL algorithm. We find that upstream learning with current CL algorithms[3] does not help the ViLT encoder generalize to multimodal and unimodal tasks in low-shot settings.

Vision-Language Tasks Figure 4 presents the low-shot transfer $\mathbb{T}_{LS}^{VL}(j)$ for all future tasks $j > i$ after training on upstream task $\mathcal{T}_{VL}^{i}$ (x-axis). We observe that low-shot transfer is always negative, implying that upstream continual learning always hurts the model's ability to learn new tasks in

[3]We do not include Adapters and Frozen-Encoder as they do not modify pre-trained ViLT's parameters.

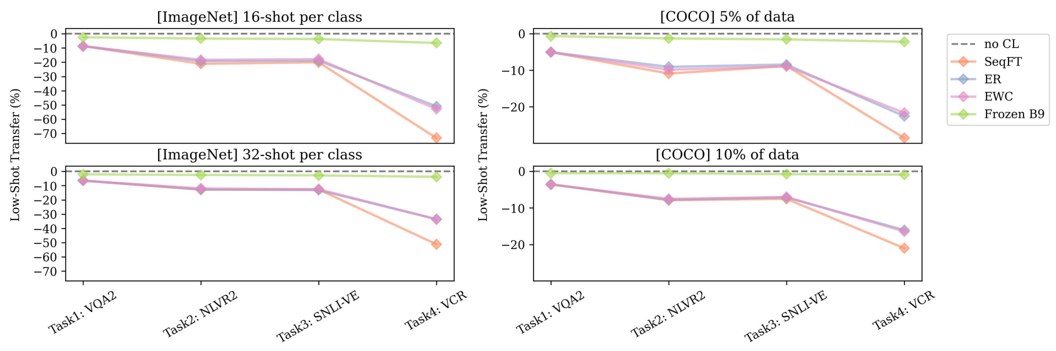


Figure 5: Low-Shot Transfer (%) comparison between different CL algorithms on downstream vision-only tasks (left: ImageNet; right: COCO). Findings on iNaturalist 2019 and Places365 are similar to ImageNet (see Supp). Generally, current CL algorithms hurt low-shot transfer compared to direct fine-tuning, with Frozen Bottom-9 being the least harmful.

low-shot settings. Since upstream learning hurts model adaptation on new multimodal tasks with full training data (Table 3), it is expected that this effect will also be reflected in the low-shot regime.

Vision-Only Tasks Figure 5 presents low-shot transfer on vision downstream tasks, using checkpoints from different upstream CL algorithms. Fine-tuning ViLT without CL performs well on vision-only tasks, achieving 65% top-1 accuracy on ImageNet-1000 with only 16 shots per class (see Supp). This performance suggests that ViLT already contains rich visual representations, making it sample efficient when transferred to vision-only tasks.

Second, CL actually *hurts* the transferability to downstream vision tasks. Among CL algorithms, Sequential Fine-tuning is the most harmful one, while freezing the bottom 9 layers causes the least degradation, almost matching direct fine tuning. This finding is consistent with previous work suggesting that bottom layers in deep models learn more general and transferable representations than upper layers [Yosinski et al., 2014, Lee et al., 2019].

Notably, upstream VQA and SNLI-VE checkpoints have a less negative effect on downstream COCO performance compared to NLVR2 and VCR. Because images from NLVR2 and VCR are more dissimilar to MS-COCO than the image sources of VQA and SNLI-VE, we hypothesize that large data distribution shifts between upstream and downstream tasks hurts transfer.

Language-Only Tasks In Figure 6, we compare the performance of two pre-trained encoders, ViLT and ViLT-BERT, on low-shot language tasks, and the effects of upstream multimodal CL on low-shot transfer when applied to both encoders.

We observe that model adaptation to language tasks is challenging. The ViLT model frequently performs only marginally better than the random baseline, regardless of the upstream algorithm. Using ViLT-BERT as the encoder achieves notably higher accuracy compared to ViLT on all tasks, indicating that strong language priors are key to low-shot language adaptation.

All upstream CL tasks improve ViLT-BERT's transferability to SST-2 except for VCR. For both SST-2 and IMDb, there are significant drops in transferability after learning VCR in the upstream phase with ViLT and ViLT-BERT, following vision-only task results showing VCR is farther out of distribution than other upstream tasks.

However, we do not observe similar trends on the three multiple-choice tasks, where CL generally hurts. We believe that current multimodal tasks do not learn complex language reasoning skills, hurting model transferability to language-only reasoning tasks.

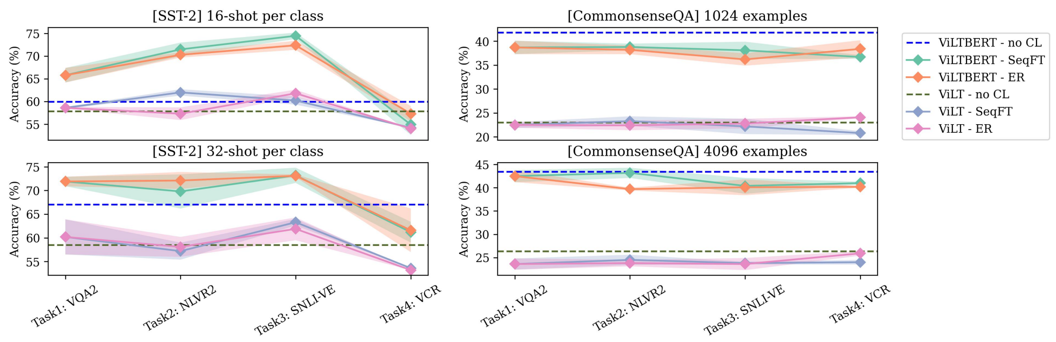


Figure 6: Comparisons between different encoders and continual learning algorithms on downstream language-only tasks (left: SST-2; right: CommonsenseQA). Note that the proposed Low-Shot Transfer metric is computed relative to the pre-trained encoder, making scores for different encoders (in this case, ViLT and ViLT-BERT) incomparable. Hence, we plot the absolute accuracy with shaded standard deviation. ViLT-BERT strictly improves absolute accuracy over ViLT in direct fine-tuning and CL settings. See Supp for comparable IMDb, HellaSwag, and PIQA results.

6 Discussion and Limitations

We propose the **Continual Learning in Multimodality Benchmark (CLiMB)** to study CL in multimodal tasks with deployment to multi- and unimodal tasks. Our experiments find that existing CL strategies do not generalize well to sequences of multimodal tasks or enable effective low-shot adaptation to downstream multi- or unimodal tasks. We hope CLiMB will allow systematic evaluation of new models and algorithms for multimodal continual learning.

CLiMB is designed to be an extensible community tool for studying tasks, model architectures, and CL algorithms. The initial CLiMB release is limited to English-only text, eliding the challenges of multi-lingual language tasks. Further, images in currently included datasets are sourced from social media, movie clips and web searches, thus excluding certain image domains, including those taken for particular needs such as descriptions for people with blindness [Gurari et al., 2018]. Such biases in a benchmark, inherited from the multi- and unimodal datasets selected, serve the needs of English-speaking, able-bodied folks as a "default." Further, we currently only perform experiments with a ViLT-based model, which directly operates on image patches, unlike other vision-language encoders which use pre-extracted object features [Chen et al., 2020, Lu et al., 2019]. However, our benchmark is model-agnostic and can be extended to models using object features.

The current CLiMB design allows for task-specific parameters and for model awareness of the task, but multi-task language modelling has seen impressive results from reframing all tasks as sequence-to-sequence problems that remove task-specific parameters [Raffel et al., 2020]. In future iterations of CLiMB, we intend to explore this task-agnostic paradigm, building further on the promising Adapters methods by learning a library of adapters that are dynamically selected based on input vision and language on a per-instance basis. Finally, the task formulations in CLiMB are mostly classification, but sequence-based vision-and-language tasks could allow the study of embodied navigation [Anderson et al., 2018] and task completion [Shridhar et al., 2020], and may be feasible in a more task-agnostic CLiMB framework.

Acknowledgments and Disclosure of Funding

This work was supported by the Laboratory for Analytic Sciences (LAS), National Security & Special Research Initiatives, and in part by DARPA under contract HR001121C0168.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2020.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv*, 2019.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. In *European Conference on Computer Vision (ECCV)*, 2020.

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

Georgios Chochlakis, Tejas Srinivasan, Jesse Thomason, and Shrikanth Narayanan. ViLT-BERT: Augmenting the Vision-and-Language Transformer with the propagation of deep language representations. *in preparation*, 2022.

Karan Desai and Justin Johnson. VirTex: Learning visual representations from textual annotations. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

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. In *International Conference on Learning Representations*, 2020.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Claudio Greco, Barbara Plank, Raquel Fernández, and Raffaella Bernardi. Psycholinguistics meets continual learning: Measuring catastrophic forgetting in visual question answering. In *Association for Computational Linguistics (ACL)*, 2019.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

Jack Hessel, Jena D Hwang, Jae Sung Park, Rowan Zellers, Chandra Bhagavatula, Anna Rohrbach, Kate Saenko, and Yejin Choi. The abduction of sherlock holmes: A dataset for visual abductive reasoning. *arXiv*, 2022.

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 (ICML)*, 2019.

Woojeong Jin, Dong-Ho Lee, Chenguang Zhu, Jay Pujara, and Xiang Ren. Leveraging visual knowledge in language tasks: An empirical study on intermediate pre-training for cross-modal knowledge transfer. In *Association for Computational Linguistics (ACL)*, 2022.

Xisen Jin, Bill Yuchen Lin, Mohammad Rostami, and Xiang Ren. Learn continually, generalize rapidly: Lifelong knowledge accumulation for few-shot learning. In *Findings of Empirical Methods in Natural Language Processing (Findings of EMNLP)*, 2021.

Wonjae Kim, Bokyung Son, and Ildoo Kim. ViLT: Vision-and-language transformer without convolution or region supervision. *International Conference on Machine Learning (ICML)*, 2021.

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.

Jaejun Lee, Raphael Tang, and Jimmy Lin. What would elsa do? freezing layers during transformer fine-tuning. *arXiv*, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The CLEAR benchmark: Continual learning on real-world imagery. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Association for Computational Linguistics (ACL)*, 2011.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *European Conference on Computer Vision (ECCV)*, 2018.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv*, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 21(140):1–67, 2020.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv*, 2022.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Neural Information Processing Systems (NeurIPS)*, 28: 91–99, 2015.

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 (IJCV)*, 115(3):211–252, 2015.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. In *International Conference on Learning Representations*, 2019.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In *Association for Computational Linguistics (ACL)*, 2019.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The iNaturalist species classification and detection dataset. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Neural Information Processing Systems (NeurIPS)*, 2016.

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

Ning Xie, Farley Lai, Derek Doran, and Asim Kadav. Visual entailment: A novel task for fine-grained image understanding. *arXiv*, 2019.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Neural Information Processing Systems (NeurIPS)*, 2014.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019a.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Association for Computational Linguistics (ACL)*, 2019b.

Rowan Zellers, Ximing Lu, Jack Hessel, Youngjae Yu, Jae Sung Park, Jize Cao, Ali Farhadi, and Yejin Choi. Merlot: Multimodal neural script knowledge models. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work?
   [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work?
   [Yes] See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)?
   [Yes] Our code and instructions can be found at `https://github.com/GLAMOR-USC/CLiMB` (mentioned on Page 1).
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)?
   [Yes] See Supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)?
   [Yes] Yes, for language-only tasks (See Figure 6), where we run experiments with three different random seeds. For other tasks, we do a single run with fixed the random seed, due to training time cost.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)?
   [Yes] See Supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets?
   [Yes] Our benchmark license is mentioned in the project's GitHub repository (linked above). Licenses of datasets included (but not distributed) are mentioned in the Supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Multimodal Task Details

Table 4 shows details about individual multimodal tasks, including hyperparameters used to train ViLT for each task, and details about how low-shot versions of each task are sampled.

For NLVR2 and SNLI-VE, where the output labels are a small number of semantically meaningful categories (True/False and Entailment/Contradiction/Neutral respectively), we sample N shots per output label to construct our low-shot training data. The 4 output labels in VCR are not semantically meaningful (since the options are interchangeable); hence, instead of sampling an equal number of training samples per label, we sample a percentage of the full training data instead. For VQAv2, the output label space is very large, and answers are not uniformly distributed across the training data, so instead of sampling N shots per output label (answer) we again sample a percentage of the full VQAv2 training data.

Task	VQAv2	NLVR2	SNLI-VE	VCR (Q $\rightarrow$ A)
		Task Details		
Task Type	Classification	Classification	Classification	Multi-Choice
Visual Input	1 Image	2 Images	1 Image	1 Image, Object boxes
Text Input	Question	Statement	Hypothesis	1 question, 4 answers
# Output Labels	3129	2	3	4
Random Score, S_R^i (%)	0.0	50.0	33.33	25.0
		Training Details/Hyperparameters		
Learning Rate	10^{-4}	10^{-4}	5×10^{-5}	10^{-4}
Weight Decay	10^{-2}	10^{-2}	10^{-2}	10^{-2}
Adam Epsilon	10^{-8}	10^{-8}	10^{-8}	10^{-8}
Num. Epochs	10	10	5	10
Batch Size	64	32	64	16
		Low-Shot Task Transformation		
Number of shots per class	-	2048	2048	-
% of training data	5%	4.74%	1.16%	5%

Table 4: Task-specific implementation details

B ViLT Model Modification Details

B.1 Applying ViLT to Multi-Choice Tasks

B.1.1 Applying ViLT to VCR

The VCR task provides object boxes, with each box corresponding to a grounded entity in the question. Unlike other pre-trained vision-language encoders [Su et al., 2019, Chen et al., 2020] that use visual features from regions-of-interest (ROIs) in the image, ViLT is designed to operate over image patches, thus making it challenging to use the object box inputs provided in the VCR task. We follow previous work [Zellers et al., 2021, Hessel et al., 2022] and draw colored boxes directly on the image corresponding to grounded references in the text. The grounded text references, *e.g.* `[person1]`, `[car1]`, are replaced with gender-neutral names for persons and object class names for all other objects. We use consistent mappings between the box colors and object names; for example, the `[person1]` object is always referenced with a green box in the image, and the name Casey in the text.

During training and inference, each possible answer a_i is paired with the question q, to form a sequence "[CLS] q [SEP] a_i". Each question-answer option is passed into the ViLT transformer, and the classifier produces a scalar score for each choice on top of the [CLS] representation. The choice with the maximum score is selected as the answer.

B.1.2 Applying ViLT to HellaSwag, PIQA, and CommonsenseQA

The inputs of language-only multiple-choice tasks consist of two parts: a sentence s (a sentence prefix in HellaSwag; a question in PIQA and CommonsenseQA), and a set of choices $A = \{a_1, a_2, ..., a_n\}$. We follow the original implementations [Zellers et al., 2019b, Bisk et al., 2020] to model these tasks, which consider different choices independently. For each choice a_i, we concatenate s and a_i with special tokens as the input: "[CLS] s [SEP] a_i [SEP]". We build the classifier, which outputs a scalar score for each choice, atop the [CLS] representation of ViLT transformer. During fine-tuning, we aggregate the scores of different choices and train the model with cross-entropy loss over the choices.

B.2 Applying ViLT to Unimodal Tasks

Sub-sampling. We conduct low-shot experiments to test the model's transferability to unimodal tasks. However, different sub-samples the training set may lead to different results. To deal with this issue, for every language-only task, we use three different random seeds for sub-sampling, leading to three different training subsets, and then report the mean and standard deviation of the accuracy scores on the full validation set. For vision-only tasks, however, we observed low variances in accuracy across three sub-samplings ($39.35 \pm 0.4\%$ on Places 365; 16-shot per class). Thus, we fix the random seed and only use a single training subset for vision-only tasks due to the computational cost.

What's the language input for vision-only tasks? For vision-only tasks, we found that simply using "This is an image." as the language input works empirically well on all tasks. While the performance could be further improved by using more informative textual inputs, we leave it as future work as this is not the primary focus of this paper.

What's the visual input for language-only tasks? For language-only tasks, to keep the visual input in-distribution, we first resize all MS-COCO training images into 384×384 and then average them into a single image (see Fig 7) as the vacuous visual input, since MS-COCO is one of the pretrained corpus of ViLT. 384 is the shorter-edge image size used by ViLT, and with the default 32×32 patch projection, it takes $(384/32) \times (384/32) = 144$ image tokens in the original implementation. We also conduct ablation studies that include two baselines: (1) not inputting any image to ViLT at all, and (2) inputting the zero-vector image instead of the average image of the COCO dataset. The first three rows in Table 5 show that inputting the average image is slightly better than the other two baselines, presenting the benefit of using an in-distribution image, even when it is vacuous.

Figure 7: The average image of the MS-COCO dataset.

Language-and-vision token reallocation. In language-only tasks, we would like to focus on the language inputs, which only accounts for 40 tokens in the original ViLT implementation, instead of the vacuous visual input, which accounts for 144 tokens. Thus, we extend the language inputs by extending the positional embeddings to a maximum of 160, which is jointly learned during fine-tuning, and decrease the image tokens by downsample the image size to 128×128, which now only takes $(128/32) \times (128/32) = 16$ image tokens. The last row in Table 5 shows that this re-allocation of language-and-vision position embedding tokens notably improves the performance on language tasks.

B.3 ViLT-BERT Implementation Details

The ViLT-BERT model is a modification of the ViLT model that uses stronger language priors. Since the ViLT Transformer was initialized using weights from the vision transformer ViT [Dosovitskiy et al., 2020], and pre-trained only on image caption datasets, the language understanding of ViLT is limited to a specific language domain of image captions, thus making it unsuitable for language-only tasks. We perform additional experiments with a ViLT-BERT model [Chochlakis et al., 2022] that replaces the language input embeddings of the ViLT Transformer with language token representations

	16-shot	32-shot	128-shot
ViLT-40 -no image	51.2 ± 0.4	54.0 ± 1.2	56.8 ± 1.2
ViLT-40 -zero	53.8 ± 0.1	54.8 ± 0.3	56.7 ± 0.7
ViLT-40 -avg	53.7 ± 0.8	55.1 ± 1.0	58.0 ± 0.6
ViLT-160 -avg	**55.9** ± 2.1	**57.8** ± 1.5	**62.3** ± 0.5

Table 5: Accuracy (%) of vacuous visual input variants on IMDb, with $N = \{16, 32, 128\}$ shot per class. $-l$ means the maximum language sequence length is l, where ViLT-160 -avg is the proposed method that reallocates the language-and-vision tokens and has fewer visual tokens than other rows.

extracted from a frozen, pre-trained BERT model. We refer to this modified approach as ViLT-BERT. ViLT-BERT has more effective language understanding, but the more general language representations could hurt its performance on vision-language tasks.

B.3.1 ViLT vs ViLT-BERT Multimodal Task Comparison

Table 6 shows a comparison of pre-trained ViLT and ViLT-BERT when directly trained on each of the upstream vision-language tasks. ViLT-BERT underperforms ViLT across all multimodal tasks.

Model	VQAv2	NLVR2	SNLI-VE	VCR
ViLT	67.70%	73.07%	76.31%	61.31%
ViLT-BERT	65.80%	65.57%	74.12%	59.46%

Table 6: Comparison of pre-trained ViLT versus ViLT-BERT when trained directly on each of the upstream multimodal tasks. ViLT-BERT consistently underperforms ViLT's accuracy.

B.3.2 ViLT vs ViLT-BERT Language-Only Task Comparison

Table 7 compares pre-trained ViLT and ViLT-BERT when directly fine-tuned on downstream language-only tasks, showing that ViLT-BERT can significantly improve the accuracy over ViLT.

Model	IMDb		SST-2	
	16	32	16	32
ViLT	55.9 ± 2.1	57.8 ± 1.5	57.8 ± 3.6	58.5 ± 7.4
ViLT-BERT	**64.8** ± 2.0	**70.0** ± 1.7	**59.9** ± 3.0	**67.0** ± 3.2

Model	HellaSwag		CommonsenseQA		PIQA	
	1024	4096	1024	4096	1024	4096
ViLT	26.5 ± 0.3	27.7 ± 0.3	23.0 ± 2.0	26.3 ± 0.5	52.0 ± 0.7	54.8 ± 0.5
ViLT-BERT	**31.7** ± 0.5	**32.9** ± 0.8	**41.8** ± 0.6	**43.4** ± 0.6	**54.6** ± 1.0	**58.0** ± 0.8

Table 7: Comparisons between ViLT and ViLT-BERT on downstream language-only tasks. Each value is the average accuracy (%) and standard deviation over 3 runs. We experiment with $\{16, 32\}$-shot per class on IMDB, SST-2 and sub-sample $\{1024, 4096\}$ training data for HellaSwag, CommonSenseQA, and PIQA. ViLT-BERT consistently achieves higher accuracy than ViLT.

Similarly, Figure 8 shows that ViLT-BERT strictly improves absolute accuracy over ViLT in direct fine-tuning and CL settings for downstream language-only tasks.

C Algorithm Implementation Details

For **Sequential Fine-tuning**, we finetune the shared encoder parameters when learning each task, whereas for **Frozen Encoder**, we keep the shared encoder frozen and only fine-tune the task-specific

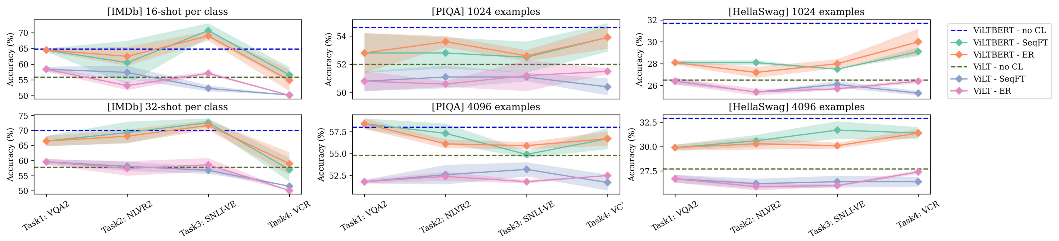


Figure 8: Comparisons between ViLT and ViLT-BERT with checkpoints from different CL algorithms on downstream language-only tasks. We conduct three runs of different training sub-samplings and plot the absolute accuracy with shaded standard deviation.

classification layers. In **Frozen Bottom-K**, the embedding parameters and the bottom K (< 12) transformer layers are frozen; we set K=9 in our experiments.

The **Experience Replay (ER)** algorithm has two hyperparameters: the percentage of each task's training samples to be stored in the memory buffer, and the frequency with which to perform a "replay step". We set these hyperparameters as 1% of training data and 100 training steps, respectively. We sample training examples for the memory buffer randomly from the training dataset; alternatives include sampling an equal number of training examples per output class.

Elastic Weight Consolidation (EWC) consists of computing the Fisher information matrix from the training data, which determines the importance of each parameter in the shared encoder. Rather than doing a full pass through the whole training data to construct the Fisher information matrix for each task, we use only 1% of the training examples. During training an upstream task, we sample one of the previous tasks and compute the L2 loss between parameter values in the current encoder and the previous task's encoder checkpoint. The L2 loss is weighted by that parameter's Fisher information and summed across all parameters. This EWC loss $\mathcal{L}_{EWC}$ is multiplied by a constant λ and added to the upstream task loss $\mathcal{L}_{task}$. We select $\lambda = 10^2$ based on a hyperparameter sweep.

Adapters add a 2-layer MLP, also known as an Adapter module, after every Self-Attention and Feed-forward layer in each Transformer block. Following the original Houlsby configuration [Houlsby et al., 2019], the first layer of each Adapter module is a downsampling layer, which reduces the dimensionality of the input features by a factor of 16, followed by a GELU activation function, and finally an upsampling layer which produces an output representation with the same dimensionality as the Adapter input.

D Full ViLT Results

D.1 Full Catastrophic Forgetting Results

In Table 8, we present full forgetting transfer numbers for all six CL algorithms, which were reported in a compact form in Figure 3a.

D.2 Full Results of Different Upstream Task Orders

Table 9a contains full results of upstream knowledge transfer $\mathbb{T}_{UK}(i)$,when the ViLT encoder sees different sequences of upstream tasks. We use Sequential Fine-tuning for all these experiments. Table 9b shows the forgetting of previous tasks, for these different upstream task orders. We previously summarized these results visually in Figure 3b.

D.3 Full Results of Low-Shot Multimodal Transfer

In Table 10, we present the full results when the CL-learned ViLT encoder, after training on the i^{th} task, is trained on future low-shot tasks $\mathcal{T}_{VL}^{LS(j)}$ for $j = \{i+1, ..., K_{VL}\}$. The first section of the

CL Algorithm: Sequential Fine-tuning

Checkpoint \ Evaluated on	Task 1 VQAv2	Task 2 NLVR2	Task 3 SNLI-VE
After training on that task	[67.79]	[72.66]	[74.89]
Task 2: NLVR2	40.97% [40.02]	-	-
Task 3: SNLI-VE	39.25% [41.18]	43.81% [62.73]	-
Task 4: VCR	63.90% [24.47]	93.74% [51.24]	89.93% [37.52]

CL Algorithm: Frozen Encoder

Checkpoint \ Evaluated on	Task 1 VQAv2	Task 2 NLVR2	Task 3 SNLI-VE
After training on that task	[58.15]	[63.66]	[69.45]
Task 2: NLVR2	-0.38% [58.37]	-	-
Task 3: SNLI-VE	-0.38% [58.37]	-0.31% [63.70]	-
Task 4: VCR	-0.38% [58.37]	-0.42% [63.72]	0.00% [69.45]

CL Algorithm: Frozen Bottom-9

Checkpoint \ Evaluated on	Task 1 VQAv2	Task 2 NLVR2	Task 3 SNLI-VE
After training on that task	[67.30]	[72.94]	[74.90]
Task 2: NLVR2	16.97% [55.90]	-	-
Task 3: SNLI-VE	21.36% [52.93]	29.32% [66.21]	-
Task 4: VCR	71.61% [19.11]	78.52% [54.93]	35.01% [60.34]

CL Algorithm: Experience Replay

Checkpoint \ Evaluated on	Task 1 VQAv2	Task 2 NLVR2	Task 3 SNLI-VE
After training on that task	[67.87]	[73.20]	[75.08]
Task 2: NLVR2	12.88% [59.13]	-	-
Task 3: SNLI-VE	12.96% [59.07]	17.10% [69.23]	-
Task 4: VCR	43.62% [38.27]	78.27% [55.04]	33.45% [61.11]

CL Algorithm: Elastic Weight Consolidation

Checkpoint \ Evaluated on	Task 1 VQAv2	Task 2 NLVR2	Task 3 SNLI-VE
After training on that task	[67.84]	[72.39]	[74.38]
Task 2: NLVR2	39.81% [40.83]	-	-
Task 3: SNLI-VE	31.52% [46.46]	25.73% [66.66]	-
Task 4: VCR	65.25% [23.58]	81.03% [54.25]	73.61% [43.34]

CL Algorithm: Adapters

Checkpoint \ Evaluated on	Task 1 VQAv2	Task 2 NLVR2	Task 3 SNLI-VE
After training on that task	[68.10]	[73.66]	[76.08]
Task 2: NLVR2	-0.01% [68.11]	-	-
Task 3: SNLI-VE	0.04% [68.07]	3.51% [72.83]	-
Task 4: VCR	0.67% [67.64]	6.48% [72.13]	0.89% [75.70]

Table 8: Full numbers for forgetting transfer $\mathbb{T}_F(j \leftarrow i)$ of previously seen tasks for each CL algorithm. We also show the transfer score $[S_{\mathcal{A}}^{j \leftarrow i}]$ when evaluated on that task after training on future task i. The first row contains task score $[S_{\mathcal{A}}^{j}]$ after originally training on j^{th} task.

table contains a comparison of ViLT's performance when directly fine-tuned on each task, when both full training data and low-shot versions of the task are available. The following sections show the low-shot transfer when upstream checkpoints, trained using four of our six CL algorithms, are used to fine-tuned on low-shot tasks. We do not perform experiments with the Frozen-Encoder and Adapter algorithms, as the encoder parameters are identical to the pre-trained checkpoint.

Directly fine-tuning pre-trained ViLT on each task

VQAv2	NLVR2	SNLI-VE	VCR
[67.70]	[73.07]	[76.31]	[61.31]

Task Order: VQAv2 → NLVR2 → SNLI-VE → VCR

Task 1	Task 2	Task 3	Task 4
VQAv2	NLVR2	SNLI-VE	VCR
0.13% [67.79]	-1.80% [72.66]	-3.33% [74.89]	-5.09% [59.47]

Task Order: SNLI-VE → VCR → VQAv2 → NLVR2

Task 1	Task 2	Task 3	Task 4
SNLI-VE	VCR	VQAv2	NLVR2
-0.07% [76.29]	-1.55% [60.75]	-6.55% [63.27]	-21.35% [67.65]

Task Order: NLVR2 → VQAv2 → VCR → SNLI-VE

Task 1	Task 2	Task 3	Task 4
NLVR2	VQAv2	VCR	SNLI-VE
0.06% [73.25]	-1.52% [66.55]	-6.03% [59.10]	-7.88% [73.07]

(a) Full knowledge transfer results with different task orders.

Task Order: VQAv2 → NLVR2 → SNLI-VE → VCR

Evaluated on / Checkpoint	Task 1 VQAv2	Task 2 NLVR2	Task 3 SNLI-VE
After training on that task	[67.79]	[72.66]	[74.89]
Task 2: NLVR2	40.97% [40.02]	-	-
Task 3: SNLI-VE	39.25% [41.18]	43.81% [62.73]	-
Task 4: VCR	63.90% [24.47]	93.74% [51.24]	89.93% [37.52]

Task Order: SNLI-VE → VCR → VQAv2 → NLVR2

Evaluated on / Checkpoint	Task 1 SNLI-VE	Task 2 VCR	Task 3 VQAv2
After training on that task	[76.29]	[60.75]	[63.27]
Task 2: VCR	84.50% [39.99]	-	-
Task 3: VQAv2	85.86% [39.40]	91.47% [28.05]	-
Task 4: NLVR2	77.56% [42.97]	86.11% [29.97]	41.94% [36.73]

Task Order: NLVR2 → VQAv2 → VCR → SNLI-VE

Evaluated on / Checkpoint	Task 1 NLVR2	Task 2 VQAv2	Task 3 VCR
After training on that task	[73.25]	[66.55]	[59.10]
Task 2: VQAv2	58.06% [59.68]	-	-
Task 3: VCR	90.63% [52.16]	68.69% [20.87]	-
Task 4: SNLI-VE	91.75% [51.90]	62.59% [24.94]	34.04% [47.51]

(b) Full forgetting results with different task orders.

Table 9: Effects of CL task order on the ViLT encoder's upstream knowledge transfer and forgetting.

D.4 Full Results of Low-Shot Unimodal Transfer

Vision-only downstream tasks. Table 11 presents the full results of vision-only tasks in absolute accuracy (%). Figure 9 plots the same results with Low-Shot Transfer (%). First, in single-task fine-tuning, we only include a single task in the upstream phase and compare the influence of different upstream tasks to downstream low-shot transfer. We found that across all vision-only downstream tasks, SNLI-VE > VQAv2 > NLVR2 > VCR, where VCR as the upstream task significantly damages the model performance. Second, current CL algorithms always *hurt* low-shot transfer compared to direct fine-tuning. Among them, Frozen Bottom-9 is the least harmful algorithm. Experience Replay and EWC perform similarly, and both are notably better than Sequential Fine-Tuning after training on VCR. In conclusion, the pretrained ViLT already achieves decent performance on low-shot vision-only classification tasks. Meanwhile, with current CL algorithms, the model does not benefit

Directly fine-tuning on each task

Training Data presented	Task 2 NLVR2	Task 3 SNLI-VE	Task 4 VCR
Full Training Data, S^i_{PT}	[73.07]	[76.31]	[61.31]
Low-Shot Transfer, $S^{LS(i)}_{PT}$	[62.46]	[65.67]	[43.23]

CL Algorithm: Sequential Fine-tuning

Low-Shot Transfer to	Task 2 NLVR2	Task 3 SNLI-VE	Task 4 VCR
After training on Task 1: VQAv2	-8.19% [61.44]	-4.51% [64.21]	-13.71% [40.73]
After training on Task 2: NLVR2	-	-14.87% [60.86]	-26.60% [38.48]
After training on Task 3: SNLI-VE	-	-	-18.71% [39.82]

CL Algorithm: Frozen Bottom-9

Low-Shot Transfer to	Task 2 NLVR2	Task 3 SNLI-VE	Task 4 VCR
After training on Task 1: VQAv2	-15.73% [60.50]	-0.87% [65.39]	-4.22% [42.46]
After training on Task 2: NLVR2	-	-0.99% [65.35]	-10.48% [41.32]
After training on Task 3: SNLI-VE	-	-	-4.00% [42.50]

CL Algorithm: Experience Replay

Low-Shot Transfer to	Task 2 NLVR2	Task 3 SNLI-VE	Task 4 VCR
After training on Task 1: VQAv2	-7.95% [61.47]	-1.76% [65.10]	-15.47% [40.41]
After training on Task 2: NLVR2	-	-10.48% [62.28]	-26.82% [38.34]
After training on Task 3: SNLI-VE	-	-	-18.38% [39.88]

CL Algorithm: Elastic Weight Consolidation

Low-Shot Transfer to	Task 2 NLVR2	Task 3 SNLI-VE	Task 4 VCR
After training on Task 1: VQAv2	-13.24% [60.81]	-2.01% [65.02]	-15.52% [40.40]
After training on Task 2: NLVR2	-	-17.53% [60.00]	-29.29% [37.89]
After training on Task 3: SNLI-VE	-	-	-22.87% [39.06]

Table 10: Full low-shot multiodal transfer results, when transferring ViLT checkpoints from upstream CL training to future multimodal tasks.

from training on more vision-and-language upstream tasks, but suffers from forgetting useful visual representations, learned in pretraining, for downstream tasks.

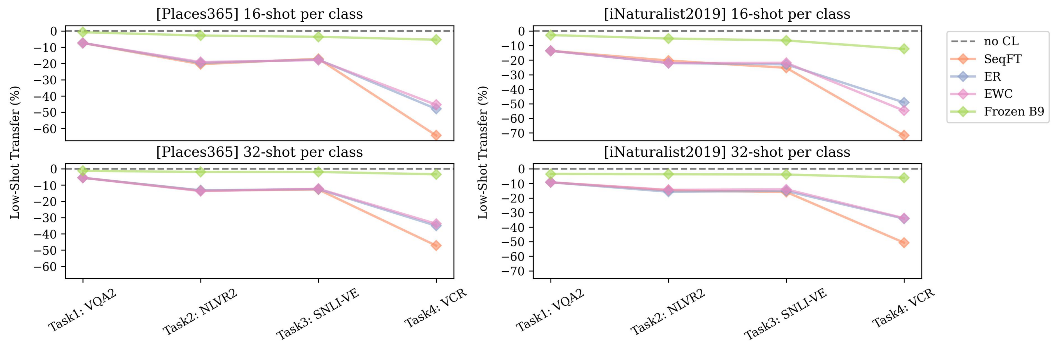


Figure 9: Low-Shot Transfer (%) comparison between different CL algorithms on downstream vision-only tasks (left: Places365; right: iNaturalist2019).

Language-only downstream tasks. Table 12 presents the full results of language-only tasks in absolute accuracy (%). First, ViLT performs poorly on language-only tasks. Similar to our findings in vision-only tasks and multimodal tasks, including VCR as one of the upstream tasks hurts the model performance on downstream tasks, most notably on SST-2 and IMDb. Although VCR is also a

Checkpoint \ Task	ImageNet		iNat2019		Places365		COCO	
	16	32	16	32	16	32	5%	10%
Direct Fine-Tuning								
ViLT	64.4	67.7	46.3	54.1	39.2	41.7	77.1	78.5
CL: Singe-Task Fine-Tuning								
After Task1: SNLI-VE	62.3	66.3	43.6	53.1	37.6	40.5	74.6	77.4
After Task1: VQAv2	58.8	63.3	40.0	49.1	36.3	39.4	73.2	75.7
After Task1: NLVR2	56.2	62.7	36.4	48.4	31.9	37.0	67.3	73.1
After Task1: VCR	25.2	45.8	10.3	34.4	17.6	26.6	60.7	66.8
CL: Sequential Fine-Tuning								
After Task2: NLVR2	59.0	50.8	36.9	46.1	31.2	36.1	68.7	72.3
After Task3: SNLI-VE	51.5	59.1	34.6	45.5	32.5	36.4	70.3	72.6
After Task4: VCR	17.3	33.1	13.1	26.7	14.0	22.0	55.1	62.0
CL: Experience Replay								
After Task2: NLVR2	52.0	59.1	36.1	45.6	31.5	36.2	70.1	72.4
After Task3: SNLI-VE	52.2	58.8	35.7	45.9	32.3	36.5	70.6	72.9
After Task4: VCR	31.6	45.0	23.6	35.6	20.4	27.1	59.7	65.9
CL: EWC								
After Task2: NLVR2	52.6	59.6	36.1	46.3	31.7	36.0	69.5	72.6
After Task3: SNLI-VE	52.9	59.2	36.2	46.5	32.2	36.6	70.2	73.0
After Task4: VCR	30.4	45.0	21.0	35.8	21.4	27.6	60.4	65.6
CL: Frozen Bottom-9								
After Task1: VQAv2	62.8	66.3	45.0	52.2	38.9	41.2	76.6	78.1
After Task2: NLVR2	62.2	66.0	43.9	52.1	38.1	40.9	76.1	78.1
After Task3: SNLI-VE	62.0	65.8	43.3	52.0	37.8	40.9	75.9	77.9
After Task4: VCR	60.2	65.1	40.6	50.8	37.1	40.3	75.4	77.8

Table 11: Comparisons between different CL algorithms on vision-only tasks. We experiment with $\{16, 32\}$-shot per class on ImageNet-1000, iNaturalist 2019, and Places 365 datasets. For COCO multi-label object detection task, we sub-sample $\{5\%, 10\%\}$ of the training data. All CL algorithms hurt the accuracy (%) compared to direct fine-tuning, while Frozen Bottom-9 is the least harmful one. Comparing different upstream tasks, SNLI-VE > VQAv2 > NLVR2 > VCR across all four downstream tasks, where VCR greatly damages the performance. Note that for Sequential Fine-Tuning, Experience Replay, and EWC, the result of "After Task1: VQAv2" is shown under Single-Task Fine-Tuning, as there are no differences between these CL algorithms in the first task.

multiple-choice commonsense reasoning task, it does not benefit HellaSwag, CommonsenseQA, and PIQA. On the other hand, continual learning sometimes improves the accuracy, especially on SST-2 and IMDb. CLiMB facilitates further investigation into these phenomena.

E Hardware and Resources Used

Our experiments were performed on an Exxact workstation containing four NVIDIA RTX 3090 GPUs. Each upstream continual learning experiment was run on a single GPU. While individual tasks took between 12 hours and 2 days to train, the entire 4-task continual learning typically took between 4 and 5 days. For downstream experiments, each (upstream checkpoint, downstream task, sample size, random seed) run took between 30 minutes to 3 hours to train, depending on the tasks.

Checkpoint \ Task	IMDb		SST-2		HellaSwag		ComQA		PIQA	
	16	32	16	32	1024	4096	1024	4096	1024	4096
Direct Fine-Tuning										
ViLT	55.9	57.8	57.8	58.5	26.5	27.7	23.0	26.3	52.0	54.8
CL: Singe-Task Fine-Tuning										
After Task1: SNLI-VE	59.0	60.5	59.8	60.6	26.8	27.3	23.3	26.5	52.7	55.2
After Task1: VQAv2	58.4	59.6	58.6	60.2	26.4	26.7	22.5	23.6	50.8	51.8
After Task1: NLVR2	56.2	56.5	55.8	58.2	26.0	27.1	21.5	25.4	52.8	54.1
After Task1: VCR	49.9	51.1	51.3	51.5	26.6	26.7	21.9	24.3	49.3	51.9
CL: Sequential Fine-Tuning										
After Task2: NLVR2	57.4	58.1	62.0	57.2	25.4	26.2	23.3	24.5	51.1	52.6
After Task3: SNLI-VE	52.3	56.8	60.2	63.3	26.1	26.4	22.2	23.8	51.1	53.2
After Task4: VCR	50.2	51.5	54.2	53.6	25.3	26.4	20.8	24.0	50.4	51.7
CL: Experience Replay										
After Task2: NLVR2	53.2	57.4	57.3	58.1	25.4	25.9	22.4	23.8	50.6	52.4
After Task3: SNLI-VE	57.1	58.6	61.8	61.9	25.7	26.0	22.8	23.6	51.2	51.8
After Task4: VCR	50.1	50.1	54.0	53.2	26.4	27.4	24.1	25.9	51.5	52.5
CL: EWC										
After Task2: NLVR2	51.0	55.1	59.7	57.0	25.2	26.4	21.6	24.1	50.0	52.1
After Task3: SNLI-VE	53.9	54.5	57.1	57.5	25.7	26.1	20.6	22.4	50.0	52.8
After Task4: VCR	49.7	49.8	51.3	51.1	25.9	27.0	22.0	23.8	51.5	52.1
CL: Frozen Bottom-9										
After Task1: VQAv2	57.4	57.9	56.5	58.9	25.8	26.9	22.5	27.3	50.3	54.4
After Task2: NLVR2	53.9	55.7	56.2	58.7	26.6	27.2	24.8	27.0	51.3	54.0
After Task3: SNLI-VE	57.8	61.6	56.8	60.7	25.7	27.1	23.5	26.6	51.1	53.0
After Task4: VCR	54.2	55.7	56.5	58.2	26.1	27.2	24.3	27.4	51.5	53.6
Random	50.0	50.0	50.0	50.0	25.0	25.0	20.0	20.0	50.0	50.0

Table 12: Comparisons between different CL algorithms on language-only tasks. Each value is the average accuracy (%) over 3 runs. We experiment with $\{16, 32\}$-shot per class on IMDB, SST-2. For HellaSwag, CommonsenseQA, and PIQA, we sub-sample $\{1024, 4096\}$ training data.