Adversarial Continual Learning

Sayna Ebrahimi[1,2], Franziska Meier[1], Roberto Calandra[1],
Trevor Darrell[2], and Marcus Rohrbach[1]

[1]Facebook AI Research, USA [2]UC Berkeley EECS, Berkeley, CA, USA
{sayna,trevor}@eecs.berkeley.edu, {fmeier,rcalandra,mrf}@fb.com

Abstract. Continual learning aims to learn new tasks without forgetting previously learned ones. We hypothesize that representations learned to solve each task in a sequence have a shared structure while containing some task-specific properties. We show that shared features are significantly less prone to forgetting and propose a novel hybrid continual learning framework that learns a disjoint representation for task-invariant and task-specific features required to solve a sequence of tasks. Our model combines architecture growth to prevent forgetting of task-specific skills and an experience replay approach to preserve shared skills. We demonstrate our hybrid approach is effective in avoiding forgetting and show it is superior to both architecture-based and memory-based approaches on class incrementally learning of a single dataset as well as a sequence of multiple datasets in image classification. Our code is available at https://github.com/facebookresearch/Adversarial-Continual-Learning.

1 Introduction

Humans can learn novel tasks by augmenting core capabilities with new skills learned based on information for a specific novel task. We conjecture that they can leverage a lifetime of previous task experiences in the form of fundamental skills that are robust to different task contexts. When a new task is encountered, these generic strategies form a base set of skills upon which task-specific learning can occur. We would like artificial learning agents to have the ability to solve many tasks sequentially under different conditions by developing task-specific and task-invariant skills that enable them to quickly adapt while avoiding *catastrophic forgetting* [24] using their memory.

One line of continual learning approaches learns a single representation with a fixed capacity in which they detect important weight parameters for each task and minimize their further alteration in favor of learning new tasks. In contrast, structure-based approaches increase the capacity of the network to accommodate new tasks. However, these approaches do not scale well to a large number of tasks if they require a large amount of memory for each task. Another stream of approaches in continual learning relies on explicit or implicit experience replay by storing raw samples or training generative models, respectively. In this paper, we propose a novel adversarial continual learning (ACL) method in which a disjoint latent space representation composed of *task-specific* or *private*

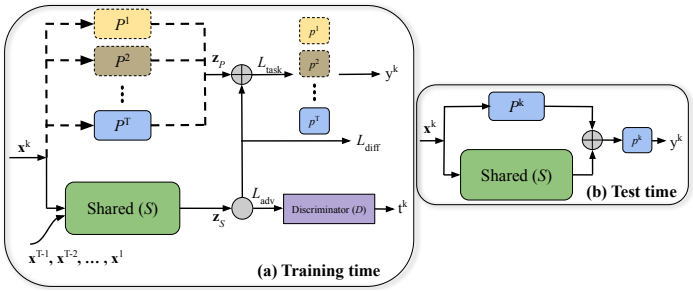

Fig. 1: Factorizing task-specific and task-invariant features in our method (ACL) while learning T sequential tasks at a time. *Left:* Shows ACL at training time where the Shared module is adversarially trained with the discriminator to generate *task-invariant* features ($\mathbf{z}_S$) while the discriminator attempts to predict task labels. Architecture growth occurs at the arrival of the k^{th} task by adding a *task-specific* modules denoted as P^k and p^k, optimized to generate orthogonal representation $\mathbf{z}_P$ to $\mathbf{z}_S$. To prevent forgetting, 1) Private modules are stored for each task and 2) A shared module which is less prone to forgetting, yet is also retrained with experience reply with a limited number of exemplars *Right:* At test time, the discriminator is removed and ACL uses the P^k module for the specific task it is evaluated on.

latent space is learned for each task and a *task-invariant* or *shared* feature space is learned for all tasks to enhance better knowledge transfer as well as better recall of the previous tasks. The intuition behind our method is that tasks in a sequence share a part of the feature representation but also have a part of the feature representation which is task-specific. The shared features are notably less prone to forgetting and the tasks-specific features are important to retain to avoid forgetting the corresponding task. Therefore, factorizing these features separates the part of the representation that forgets from that which does not forget. To disentangle the features associated with each task, we propose a novel adversarial learning approach to enforce the shared features to be task-invariant and employ orthogonality constraints [30] to enforce the shared features to not appear in the task-specific space.

Once factorization is complete, minimizing forgetting in each space can be handled differently. In the task-specific latent space, due to the importance of these features in recalling the task, we freeze the private module and add a new one upon finishing learning a task. The shared module, however, is significantly less susceptible to forgetting and we only use the replay buffer mechanism in this space to the extend that factorization is not perfect, i.e., when tasks have little overlap or have high domain shift in between, using a tiny memory containing samples stored from prior tasks will help with better factorization and hence higher performance. We empirically found that unlike other memory-based methods in which performance increases by increasing the samples from prior tasks, our model requires a very tiny memory budget beyond which its perfor-

mance remains constant. This alleviates the need to use old data, as in some applications it might not be possible to store a large amount of data if any at all. Instead, our approach leaves room for further use of memory, if available and need be, for architecture growth. Our approach is simple yet surprisingly powerful in not forgetting and achieves state-of-the-art results on visual continual learning benchmarks such as MNIST, CIFAR100, Permuted MNIST, miniImageNet, and a sequence of 5 tasks.

2 Related Work

2.1 Continual learning

The existing continual learning approaches can be broadly divided into three categories: memory-based, structure-based, and regularization-based methods.

Memory-based methods: Methods in this category mitigate forgetting by relying on storing previous experience explicitly or implicitly wherein the former raw samples [6, 21, 28, 26, 27] are saved into the memory for *rehearsal* whereas in the latter a generative model such as a GAN [32] or an autoencoder [16] synthesizes them to perform *pseudo-rehearsal*. These methods allow for simultaneous multi-task learning on i.i.d. data which can significantly reduce forgetting. A recent study on tiny episodic memories in CL [7] compared methods such as GEM [21], A-GEM [6], MER [27], and ER-RES [7]. Similar to [27], for ER-RES they used reservoir sampling using a single pass through the data. Reservoir sampling [39] is a better sampling strategy for long input data compared to random selection. In this work, we explicitly store raw samples into a very tiny memory used for replay buffer and we differ from prior work by how these stored examples are used by specific parts of our model (discriminator and shared module) to prevent forgetting in the features found to be shared across tasks.

Structure-based methods: These methods exploit modularity and attempt to localize inference to a subset of the network such as columns [29], neurons [11, 41], a mask over parameters [23, 31]. The performance on previous tasks is preserved by storing the learned module while accommodating new tasks by augmenting the network with new modules. For instance, Progressive Neural Nets (PNNs) [29] statically grow the architecture while retaining lateral connections to previously frozen modules resulting in guaranteed zero forgetting at the price of quadratic scale in the number of parameters. [41] proposed dynamically expandable networks (DEN) in which, network capacity grows according to tasks *relatedness* by splitting/duplicating the most important neurons while time-stamping them so that they remain accessible and re-trainable at all time. This strategy despite introducing computational cost is inevitable in continual learning scenarios where a large number of tasks are to be learned and a fixed capacity cannot be assumed.

Regularization methods: In these methods [18, 42, 1, 25, 10], the learning capacity is assumed fixed and continual learning is performed such that the change in parameters is controlled and reduced or prevented if it causes performance downgrade on prior tasks. Therefore, for parameter selection, there has

to be defined a *weight importance* measurement concept to prioritize parameter usage. For instance, inspired by Bayesian learning, in elastic weight consolidation (EWC) method [18] important parameters are those to have the highest in terms of the Fisher information matrix. HAT [31] learns an attention mask over *important* parameters. Authors in [10] used per-weight uncertainty defined in Bayesian neural networks to control the change in parameters. Despite the success gained by these methods in maximizing the usage of a fixed capacity, they are often limited by the number of tasks.

2.2 Adversarial learning

Adversarial learning has been used for different problems such as generative models [13], object composition [2], representation learning [22], domain adaptation [36], active learning [34], etc. The use of an adversarial network enables the model to train in a fully-differentiable manner by adjusting to solve the *minimax* optimization problem [13]. Adversarial learning of the latent space has been extensively researched in domain adaptation [14], active learning [34], and representation learning [17, 22]. While previous literature is concerned with the case of modeling single or multiple tasks at once, here we extend this literature by considering the case of continuous learning where multiple tasks need to be learned in a sequential manner.

2.3 Latent Space Factorization

In the machine learning literature, *multi-view* learning, aims at constructing and/or using different views or modalities for better learning performances [3, 40]. The approaches to tackle multi-view learning aim at either maximizing the mutual agreement on distinct views of the data or focus on obtaining a latent subspace shared by multiple views by assuming that the input views are generated from this latent subspace using Canonical correlation analysis and clustering [8], Gaussian processes [33], etc. Therefore, the concept of factorizing the latent space into *shared* and *private* parts has been extensively explored for different data modalities. Inspired by the practicality of factorized representation in handling different modalities, here we factorize the latent space learned for different tasks using adversarial learning and orthogonality constraints [30].

3 Adversarial Continual learning (ACL)

We consider the problem of learning a sequence of T data distributions denoted as $\mathcal{D}^{tr} = \{\mathcal{D}_1^{tr}, \cdots, \mathcal{D}_T^{tr}\}$, where $\mathcal{D}_k^{tr} = \{(\mathbf{X}_i^k, \mathbf{Y}_i^k, \mathbf{T}_i^k)_{i=1}^{n_k}\}$ is the data distribution for task k with n sample tuples of input ($\mathbf{X}^k \in \mathcal{X}$), output label ($\mathbf{Y}^k \in \mathcal{Y}$), and task label ($\mathbf{T}^k \in \mathcal{T}$). The goal is to sequentially learn the model $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ for each task that can map each task input to its target output while maintaining its performance on all prior tasks. We aim to achieve this by learning a disjoint latent space representation composed of a *task-specific* latent space for each task and a

task-invariant feature space for all tasks to enhance better knowledge transfer as well as better catastrophic forgetting avoidance of prior knowledge. We mitigate catastrophic forgetting in each space differently. For the *task-invariant* feature space, we assume a limited memory budget of $\mathcal{M}^k$ which stores m samples $x_{i=1\cdots m} \sim \mathcal{D}^{tr}_{j=1\cdots k-1}$ from every single task prior to k.

We begin by learning f^k_θ as a mapping from $\mathbf{X}^k$ to $\mathbf{Y}^k$. For C-way classification task with a cross-entropy loss, this corresponds to

$$\mathcal{L}_{\text{task}}(f^k_\theta, \mathbf{X}^k, \mathbf{Y}^k, \mathcal{M}^k) = -\mathbb{E}_{(x^k,y^k)\sim(\mathbf{X}^k,\mathbf{Y}^k)\cup\mathcal{M}^k} \sum_{c=1}^{C} \mathbb{1}_{[c=y^k]} \log(\sigma(f^k_\theta(x^k))) \quad (1)$$

where σ is the softmax function and the subscript $i = \{1, \cdots, n_t\}$ is dropped for simplicity. In the process of learning a sequence of tasks, an ideal f^k is a model that maps the inputs to two independent latent spaces where one contains the shared features among all tasks and the other remains private to each task. In particular, we would like to disentangle the latent space into the information shared across all tasks ($\mathbf{z}_S$) and the independent or private information of each task ($\mathbf{z}_P$) which are as distinct as possible while their concatenation followed by a task-specific head outputs the desired targets.

To this end, we introduce a mapping called Shared ($S_{\theta_S} : \mathcal{X} \to \mathbf{z}_S$) and train it to generate features that fool an adversarial discriminator D. Conversely, the adversarial discriminator ($D_{\theta_D} : \mathbf{z}_S \to \mathcal{T}$) attempts to classify the generated features by their task labels ($\mathbf{T}^{k\in\{0,\cdots,T\}}$) . This is achieved when the discriminator is trained to maximize the probability of assigning the correct task label to generated features while simultaneously S is trained to confuse the discriminator by minimizing $\log(D(S(x^k)))$. This corresponds to the following T-way classification cross-entropy adversarial loss for this minimax game

$$\mathcal{L}_{\text{adv}}(D, S, \mathbf{X}^k, \mathbf{T}^k, \mathcal{M}^k) = \min_S \max_D \sum_{k=0}^{T} \mathbb{1}_{[k=t^k]} \log\left(D\left(S\left(x^k\right)\right)\right) . \quad (2)$$

Note that the extra label zero is associated with the 'fake' task label paired with randomly generated noise features of $\mathbf{z}'_S \sim \mathcal{N}(\mu, \sum)$. In particular, we use adversarial learning in a different regime that appears in most works related to generative adversarial networks [13] such that the generative modeling of input data distributions is not utilized here because the ultimate task is to learn a discriminative representation.

To facilitate training S, we use the Gradient Reversal layer [12] that optimizes the mapping to maximize the discriminator loss directly ($\mathcal{L}_{\text{task}_S} = -\mathcal{L}_{\text{D}}$). In fact, it acts as an identity function during forward propagation but negates its inputs and reverses the gradients during back propagation. The training for S and D is complete when S is able to generate features that D can no longer predict the correct task label for leading $\mathbf{z}_S$ to become as task-invariant as possible. The private module ($P_{\theta_P} : \mathcal{X} \to \mathbf{z}_P$), however, attempts to accommodate the task-invariant features by learning merely the features that are specific to the

task in hand and do not exist in $\mathbf{z}_S$. We further factorize $\mathbf{z}_S$ and $\mathbf{z}_P$ by using orthogonality constraints introduced in [30], also known as "difference" loss in the domain adaptation literature [4], to prevent the shared features between all tasks from appearing in the private encoded features. This corresponds to

$$\mathcal{L}_{\text{diff}}(S, P, \mathbf{X}^k, \mathcal{M}^k) = \sum_{k=1}^{T} ||(S(x^k))^{\mathrm{T}} P^k(x^k)||_F^2, \tag{3}$$

where $||\cdot||_F$ is the Frobenius norm and it is summed over the encoded features of all P modules encoding samples for the current tasks and the memory.

Final output predictions for each task are then predicted using a task-specific multi-layer perceptron head which takes $\mathbf{z}_P$ concatenated with $\mathbf{z}_S$ ($\mathbf{z}_P \oplus \mathbf{z}_S$) as an input. Taken together, these loss form the complete objective for ACL as

$$\mathcal{L}_{\text{ACL}} = \lambda_1 \mathcal{L}_{\text{adv}} + \lambda_2 \mathcal{L}_{\text{task}} + \lambda_3 \mathcal{L}_{\text{diff}}, \tag{4}$$

where λ_1, λ_2, and λ_3 are regularizers to control the effect of each component. The full algorithm for ACL is given in Alg. 1 in the appendix.

3.1 Avoiding forgetting in ACL

Catastrophic forgetting occurs when a representation learned through a sequence of tasks changes in favor of learning the current task resulting in performance downgrade on previous tasks. The main insight to our approach is decoupling the conventional *single* representation learned for a sequence of tasks into two parts: a part that *must not change* because it contains task-specific features without which complete performance retrieval is not possible, and a part that is *less prone to change* as it contains the core structure of all tasks.

To *fully prevent* catastrophic forgetting in the first part (private features), we use *compact* modules that can be stored into memory. If factorization is successfully performed, the second part remains highly immune to forgetting. However, we empirically found that when disentanglement cannot be fully accomplished either because of the little overlap or large domain shift between the tasks, using a tiny replay buffer containing few samples for old data can be beneficial to retain high ACC values as well as mitigating forgetting.

3.2 Evaluation metrics

After training for each new task, we evaluate the resulting model on all prior tasks. Similar to [21, 10], to measure ACL performance we use ACC as the average test classification accuracy across all tasks. To measure forgetting we report backward transfer, BWT, which indicates how much learning new tasks has influenced the performance on previous tasks. While BWT < 0 directly reports *catastrophic forgetting*, BWT > 0 indicates that learning new tasks has helped with the preceding tasks.

$$\text{BWT} = \frac{1}{T-1} \sum_{i=1}^{T-1} R_{T,i} - R_{i,i}, \quad \text{ACC} = \frac{1}{T} \sum_{i=1}^{T} R_{T,i} \tag{5}$$

where $R_{n,i}$ is the test classification accuracy on task i after sequentially finishing learning the n^{th} task. We also compare methods based on the memory used either in the network architecture growth or replay buffer. Therefore, we convert them into memory size assuming numbers are 32-bit floating point which is equivalent to 4bytes.

4 Experiments

In this section, we review the benchmark datasets and baselines used in our evaluation as well as the implementation details.

4.1 ACL on Vision Benchmarks

Datasets: We evaluate our approach on the commonly used benchmark datasets for T-split class-incremental learning where the entire dataset is divided into T disjoint susbsets or tasks. We use common image classification datasets including **5-Split MNIST** and **Permuted MNIST** [20], previously used in [25, 42, 10], **20-Split CIFAR100** [19] used in [42, 21, 6], and **20-Split miniImageNet** [38] used in [7, 43]. We also benchmark ACL on a sequence of **5-Datasets** including **SVHN**, **CIFAR10**, **not-MNIST**, **Fashion-MNIST** and, **MNIST** and report average performance over multiple random task orderings. Dataset statistics are given in Table 5a in the appendix. No data augmentation of any kind has been used in our analysis.
Baselines: From the prior work, we compare with state-of-the-art approaches in all the three categories described in Section 2 including Elastic Weight Consolidation (EWC) [18], Progressive neural networks (PNNs) [29], and Hard Attention Mask (HAT) [31] using implementations provided by [31] unless otherwise stated. For memory-based methods including A-GEM, GEM, and ER-RES, for Permuted MNIST, 20-Split CIFAR100, and 20-Split miniImageNet, we relied on the implementation provided by [7], but changed the experimental setting from single to multi-epoch and without using 3 Tasks for cross-validation for a more fair comparison against ACL and other baselines. On Permuted MNIST results for SI [42] are reported from [31], for VCL [25] those are obtained using their original provided code, and for Uncertainty-based CL in Bayesian framework (UCB) [10] are directly reported from the paper. We also perform fine-tuning, and joint training. In fine-tuning (ORD-FT), an ordinary single module network without the discriminator is continuously trained without any forgetting avoidance strategy in the form of experience replay or architecture growth. In joint training with an ordinary network (ORD-JT) and our ACL setup (ACL-JT) we learn all the tasks jointly in a multitask learning fashion using the entire dataset at once which serves as the upper bound for average accuracy on all tasks, as it does not adhere to the continual learning scenario.
Implementation details: For all ACL experiments except for Permuted MNIST and 5-Split MNIST we used a reduced AlexNet [15] architecture as the backbone for S and P modules for a fair comparison with the majority of our baselines.

However, ACL can be also used with more sophisticated architectures (see our code repository for implementation of ACL with reduced ResNet18 backbone). However, throughout this paper, we only report our results using AlexNet. The architecture in S is composed of 3 convolutional and 4 fully-connected (FC) layers whereas P is only a convolutional neural network (CNN) with similar number of layers and half-sized kernels compared to those used in S. The private head modules (p) and the discriminator are all composed of a small 3-layer perceptron. Due to the differences between the structure of our setup and a regular network with a single module, we used a similar CNN structure to S followed by larger hidden FC layers to match the total number of parameters throughout our experiments with our baselines for fair comparisons. For 5-Split MNIST and Permuted MNIST where baselines use a two-layer perceptron with 256 units in each and ReLU nonlinearity, we used a two-layer perceptron of size 784×175 and 175×128 with ReLU activation in between in the shared module and a single-layer of size 784×128 and ReLU for each P. In each head, we also used an MLP with layers of size 256 and 28, ReLU activations, and a 14-unit softmax layer. In all our experiments, no pre-trained model is used. We used stochastic gradient descent in a multi-epoch setting for ACL and all the baselines.

5 Results and Discussion

In the first set of experiments, we measure ACC, BWT, and the memory used by ACL and compare it against state-of-the-art methods with or without memory constraints on 20-Split miniImageNet. Next, we provide more insight and discussion on ACL and its component by performing an ablation study and visualizations on this dataset. In Section 6, we evaluate ACL on a more difficult continual learning setting where we sequentially train on 5 different datasets. Finally, in section (7), we demonstrate the experiments on sequentially learning single datasets such as 20-Split CIFAR100 and MNIST variants.

5.1 ACL Performance on 20-Split miniImageNet

Starting with 20-Split miniImageNet, we split it in 20 tasks with 5 classes at a time. Table 1a shows our results obtained for ACL compared to several baselines. We compare ACL with HAT as a regularization based method with no experience replay memory dependency that achieves ACC=59.45 ± 0.05 with BWT=-0.04 ± 0.03%. Results for the memory-based methods of ER-RES and A-GEM are re(produced) by us using the implementation provided in [7] by applying modifications to the network architecture to match with ACL in the backbone structure as well as the number of parameters. We only include A-GEM in Table 1a which is only a faster algorithm compared to its precedent GEM with identical performance.

Table 1: CL results on 20-Split miniImageNet measuring ACC (%), BWT (%), and Memory (MB). (**) denotes that methods do not adhere to the continual learning setup: ACL-JT and ORD-JT serve as the upper bound for ACC for ACL/ORD networks, respectively. (*) denotes result is re(produced) by us using the original provided code. (†) denotes result is obtained using the re-implementation setup by [31]. BWT of Zero indicates the method is zero-forgetting guaranteed. (b) Cumulative ablation study of ACL on miniImageNet where P: private modules, S: shared module, D: discriminator, $\mathcal{L}_{\text{diff}}$: orthogonality constraint, and RB: replay buffer memory of one sample per class. All results are averaged over 3 runs and standard deviation is given in parentheses

(a)

Method	**ACC%**	**BWT%**	Arch (MB)	Replay Buffer (MB)
HAT* [31]	59.45(0.05)	-0.04(0.03)	123.6	-
PNN † [29]	58.96(3.50)	Zero	588	-
ER-RES* [7]	57.32(2.56)	-11.34(2.32)	102.6	110.1
A-GEM* [6]	52.43(3.10)	-15.23(1.45)	102.6	110.1
ORD-FT	28.76(4.56)	-64.23(3.32)	37.6	-
ORD-JT**	69.56(0.78)	-	5100	-
ACL-JT**	66.89(0.32)	-	5100	-
ACL (Ours)	**62.07(0.51)**	**0.00(0.00)**	113.1	8.5

(b)

#	S	P	D	$\mathcal{L}_{\text{diff}}$	RB	**ACC%**	**BWT%**
1	x					21.19(4.43)	-60.10(4.14)
2		x				29.09(5.67)	Zero
3	x		x			32.82(2.71)	-28.67(3.61
4	x	x		x		49.13(3.45)	-3.99(0.42)
5	x	x				50.15(1.41)	-14.32(2.34)
6	x	x			x	51.19(1.98)	-9.12(2.98)
7	x	x		x	x	52.07(2.49)	-0.01(0.01)
8	x	x	x			55.72(1.42)	-0.12(0.34)
9	x	x	x	x		57.66(1.44)	-3.71(1.31)
10	x	x	x		x	60.28(0.52)	0.00(0.00)
11	x	x	x	x	x	62.07(0.51)	0.00(0.00)

A-GEM and ER-RES use an architecture with 25.6M parameters (102.6MB) along with storing 13 images of size $(84 \times 84 \times 3)$ per class (110.1MB) resulting in total memory size of 212.7MB. ACL is able to outperform all baselines in ACC=$\mathbf{62.07 \pm 0.51}$, BWT=$\mathbf{0.00 \pm 0.00}$, using total memory of 121.6MB for architecture growth (113.1MB) and storing 1 sample per class for replay buffer

(8.5MB). In our ablation study in Section 5.2, we will show our performance without using replay buffer for this dataset is ACC=57.66 ± 1.44. However, ACL is able to overcome the gap by using only one image per class (5 per task) to achieve ACC=$\mathbf{62.07 \pm 0.51}$ without the need to have a large buffer for old data in learning datasets like miniImagenet with diverse sets of classes.

Table 2: Comparison of the effect of the replay buffer size between ACL and other baselines including A-GEM [6], and ER-RES [7] on 20-Split miniImageNet where unlike the baselines, ACL's performance remains unaffected by the increase in number of samples stored per class as discussed in 5.2. The results from this table are used to generate Fig. 2 in the appendix.

Samples per class		**1**	**3**	**5**	**13**
A-GEM[6]		45.14(3.42)	49.12(4.69)	50.24(4.56)	52.43(3.10)
ER-RES[7]		40.21(2.68)	46.87(4.51)	53.45(3.45)	57.32(2.56)
ACL (ours)	ACC	62.07(0.51)	61.80(0.50)	61.69(0.61)	61.33(0.40)
	BWT	0.00(0.00)	0.01(0.00)	0.01(0.00)	-0.01(0.02)

5.2 Ablation Studies on 20-Split miniImageNet

We now analyze the major building blocks of our proposed framework including the discriminator, the shared module, the private modules, replay buffer, and the *difference* loss on the miniImagenet dataset. We have performed a complete cumulative ablation study for which the results are summarized in Table 1b and are described as follows:

Shared and private modules: Using only a shared module without any other ACL component (ablation #1 in Table 1b) yields the lowest ACC of 21.19 ± 4.43 as well as the lowest BWT performance of -60.10 ± 4.14 while using merely private modules (ablation #2) obtains a slightly better ACC of 29.05 ± 5.67 and a zero-guaranteed forgetting by definition. However, in both scenarios the ACC achieved is too low considering the random chance being 20% which is due to the small size of networks used in S and P.

Discriminator and orthogonality constraint ($\mathcal{L}_{\text{diff}}$): The role of adversarial training or presence of D on top of S and P can be seen by comparing the ablations #8 and #5 where in the latter D, as the only disentanglement mechanism, is eliminated. We observe that ACC is improved from 50.15 ± 1.41 to $55.72 \pm 1.42\%$ and BWT is increased from -14.32 ± 2.34 to -$0.12 \pm 0.34\%$. On the other hand, the effect of orthogonality constraint as the only factorization mechanism is shown in ablation #4 where the $\mathcal{L}_{\text{diff}}$ can not improve the ACC performance, but it increases BWT form -14.32 ± 2.34 to -3.99 ± 0.42. Comparing ablations #8 and #4 shows the importance of adversarial training in factorizing the latent spaces versus orthogonality constraint if they were to be used individually. To compare the role of adversarial and diff losses in the presence of replay buffer (RB), we can compare #7 and #10 in which the D and $\mathcal{L}_{\text{diff}}$ are ablated,

respectively. It appears again that D improves ACC more than $\mathcal{L}_{\text{diff}}$ by reaching ACC=60.28 ± 0.52 whereas $\mathcal{L}_{\text{diff}}$ can only achieve ACC=52.07 ± 2.49. However, the effect of D and $\mathcal{L}_{\text{diff}}$ on BWT is nearly the same.

Replay buffer: Here we explore the effect of adding the smallest possible memory replay to ACL, i. e., storing one sample per class for each task. Comparing ablation #9 and the most complete version of ACL (#11) shows that adding this memory improves both the ACC and BWT by 4.41% and 3.71%, respectively. We also evaluated ACL using more samples in the memory. Table 2 shows that unlike A-GEM and ER-RES approaches in which performance increases with more episodic memory, in ACL, ACC remains nearly similar to its highest performance. Being insensitive to the amount of old data is a remarkable feature of ACL, not because of the small memory it consumes, but mainly due to the fact that access to the old data might be prohibited or very limited in some real-world applications. Therefore, for a fixed allowed memory size, a method that can effectively use it for architecture growth can be considered as more practical for such applications.

6 ACL Performance on a sequence of 5-Datasets

In this section, we present our results for continual learning of 5 tasks using ACL in Table 3b. Similar to the previous experiment we look at both ACC and BWT obtained for ACL, finetuning as well as UCB as our baseline. Results for this sequence are averaged over 5 random permutations of tasks and standard deviations are given in parenthesis. CL on a sequence of datasets has been previously performed by two regularization based approaches of UCB and HAT where UCB was shown to be superior [10]. With this given sequence, ACL is able to outperform UCB by reaching ACC=78.55($\pm$0.29) and BWT=-0.01 using only half of the memory size and also no replay buffer. In Bayesian neural networks such as UCB, there exists double number of parameters compared to a regular model representing mean and variance of network weights. It is very encouraging to see that ACL is not only able to continually learn on a single dataset, but also across diverse datasets.

7 Additional Experiments

20-Split CIFAR100: In this experiment we incrementally learn CIFAR100 in 5 classes at a time in 20 tasks. As shown in Table 3, HAT is the most competitive baseline, although it does not depend on memory and uses 27.2MB to store its architecture in which it learns task-based attention maps reaching ACC=76.96 ± 1.23%. PNN uses 74.7MB to store the lateral modules to the memory and guarantees zero forgetting. Results for A-GEM, and ER-Reservoir are re(produced) by us using a CNN similar to our shared module architecture. We use fully connected layers with more number of neurons to compensate for the remaining number of parameters reaching 25.4MB of memory. We also

stored 13 images per class (1300 images of size $(32 \times 32 \times 3)$ in total) which requires 16.0MB of memory. However, ACL achieves ACC=$(\mathbf{78.08 \pm 1.25})$% with BWT=$\mathbf{0.00 \pm 0.01}$)% using only 25.1MB to grow private modules with 167.2K parameters (0.6MB) without using memory for replay buffer which is mainly due to the overuse of parameters for CIFAR100 which is considered as a relevantly 'easy' dataset with all tasks (classes) sharing the same data distribution. Disentangling shared and private latent spaces, prevents ACL from using redundant parameters by only storing task-specific parameters in P and p modules. In fact, as opposed to other memory-based methods, instead of starting from a large network and using memory to store samples, which might not be available in practice due to confidentiality issues (*e.g.* medical data), ACL uses memory to gradually add small modules to accommodate new tasks and relies on knowledge transfer through the learned shared module. The latter is what makes ACL to be different than architecture-based methods such as PNN where the network grows by the entire *column* which results in using a highly disproportionate memory to what is needed to learn a new task with.

Permuted MNIST: Another popular variant of the MNIST dataset in CL literature is Permuted MNIST where each task is composed of randomly permuting pixels of the entire MNIST dataset. To compare against values reported in prior work, we particularly report on a sequence of $T = 10$ and $T = 20$ tasks with ACC, BWT, and memory for ACL and baselines. To further evaluate ACL's ability in handling more tasks, we continually learned up to 40 tasks. As shown in Table 4 in the appendix, among the regularization-based methods, HAT achieves the highest performance of 91.6% [31] using an architecture of size 1.1MB. Vanilla VCL improves by 7% in ACC and 6.5% in BWT using a K-means core-set memory size of 200 samples per task (6.3MB) and an architecture size similar to HAT. PNN appears as a strong baseline achieving ACC=93.5% with guaranteed zero forgetting. Finetuning (ORD-FT) and joint training (ORD-JT) results for an ordinary network, similar to EWC and HAT (a two-layer MLP with 256 units and ReLU activations), are also reported as reference values for lowest BWT and highest achievable ACC, respectively. ACL achieves the highest accuracy among all baselines for both sequences of 10 and 20 equal to ACC=98.03 ± 0.01 and ACC=97.81 ± 0.03, and BWT=-0.01% BWT=0%, respectively which shows that performance of ACL drops only by 0.2% as the number of tasks doubles. ACL also remains efficient in using memory to grow the architecture compactly by adding only 55K parameters (0.2MB) for each task resulting in using a total of 2.4MB and 5.0MB when $T = 10$ and $T = 20$, respectively for the entire network including the shared module and the discriminator. We also observed that the performance of our model does not change as the number of tasks increases to 30 and 40 if each new task is accommodated with a new private module. We did not store old data and used memory only to grow the architecture by 55K parameters (0.2MB).

5-Split MNIST: As the last experiment in this section, we continually learn $0-9$ MNIST digits by following the conventional pattern of learning 2 classes over 5 sequential tasks [25, 42, 10]. As shown in Table 6 in the appendix, we compare

Table 3: CL results on 20-Split CIFAR100 measuring ACC (%), BWT (%), and Memory (MB). (**) denotes that methods do not adhere to the continual learning setup: ACL-JT and ORD-JT serve as the upper bound for ACC for ACL/ORD networks, respectively. (*) denotes result is obtained by using the original provided code. (†) denotes result is obtained using the re-implementation setup by [31]. (^{o}) denotes result is reported by [7]. BWT of Zero indicates the method is zero-forgetting guaranteed. All results are averaged over 3 runs and standard deviation is given in parentheses.

(a) 20-Split CIFAR100

Method	**ACC%**	**BWT%**	Arch (MB)	Replay Buffer (MB)
HAT * [31]	76.96(1.23)	0.01(0.02)	27.2	-
PNN† [29]	75.25(0.04)	Zero	93.51	-
A-GEM^o [6]	54.38(3.84)	-21.99(4.05)	25.4	16
ER-RES^o [7]	66.78(0.48)	-15.01(1.11)	25.4	16
ORD-FT	34.71(3.36)	-48.56(3.17)	27.2	-
ORD-JT**	78.67(0.34)	-	764.5	-
ACL-JT**	79.91(0.05)	-	762.6	-
ACL (Ours)	**78.08(1.25)**	**0.00(0.01)**	25.1	-

(b) Sequence of 5 Datasets

Method	**ACC%**	**BWT%**	Arch (MB)	Replay Buffer (MB)
UCB * [10]	76.34(0.12)	−1.34(0.04)	32.8	-
ORD-FT	27.32(2.41)	-42.12(2.57)	16.5	-
ACL (Ours)	**78.55(0.29)**	**−0.01(0.15)**	16.5	-

ACL with regularization-based methods with no memory dependency (EWC, HAT, UCB, Vanilla VCL) and methods relying on memory only (GEM), and VCL with K-means Core-set (VCL-C) where 40 samples are stored per task. ACL reaches ACC=($\mathbf{99.76 \pm 0.03}$)% with zero forgetting outperforming UCB with ACC=99.63% which uses nearly 40% more memory size. In this task, we only use architecture growth (no experience replay) where 54.3K private parameters are added for each task resulting in memory requirement of 1.6MB to store all private modules. Our core architecture has a total number of parameters of 420.1K. We also provide naive finetuning results for ACL and a regular single-module network with (268K) parameters (1.1MB). Joint training results for the regular network (ORD-JT) is computed as ACC=99.89 ± 0.01 for ACL which requires 189.3MB for the entire dataset as well as the architecture.

Table 4: CL results on Permuted MNIST. measuring ACC (%), BWT (%), and Memory (MB). (**) denotes that methods do not adhere to the continual learning setup: ACL-JT and ORD-JT serve as the upper bound for ACC for ACL/ORD networks, respectively. (*) denotes result is obtained by using the original provided code. (‡) denotes result reported from original work. (^{o}) denotes the results reported by [7] and (^{oo}) denotes results are reported by [31]; T shows the number of tasks. Note the difference between BWT of Zero and 0.00 where the former indicates the method is zero-forgetting guaranteed by definition and the latter is computed using Eq. 5. All results are averaged over 3 runs, the standard deviation is provided in parenthesis.

Method	**ACC%**	**BWT%**	Arch (MB)	Replay Buffer (MB)
EWC^{oo} [18] (T=10)	88.2	-	1.1	-
HAT‡ [31] (T=10)	97.4	-	2.8	-
UCB‡ [10] (T=10)	91.44(0.04)	-0.38(0.02)	2.2	-
VCL*[25] (T=10)	88.80(0.23)	-7.90(0.23)	1.1	-
VCL-C* [25](T=10)	95.79(0.10)	-1.38(0.12)	1.1	6.3
PNN^o [29] (T=20)	93.5(0.07)	Zero	N/A	-
ORD-FT (T=10)	44.91(6.61)	-53.69(1.91)	1.1	-
ORD-JT** (T=10)	96.03(0.02)	-	189.3	-
ACL-JT** (T=10)	98.45(0.02)	-	194.4	-
ACL (Ours) (T=10)	**98.03(0.01)**	-0.01(0.01)	2.4	-
ACL (Ours) (T=20)	**97.81(0.03)**	0.00(0.00)	5.0	-
ACL (Ours) (T=30)	**97.81(0.03)**	0.00(0.00)	7.2	-
ACL (Ours) (T=40)	**97.80(0.02)**	0.00(0.00)	9.4	-

8 Conclusion

In this work, we proposed a novel hybrid continual learning algorithm that factorizes the representation learned for a sequence of tasks into *task-specific* and *task-invariant* features where the former is important to be fully preserved to avoid forgetting and the latter is empirically found to be remarkably less prone to forgetting. The novelty of our work is that we use adversarial learning along with orthogonality constraints to disentangle the shared and private latent representations which results in compact private modules that can be stored into memory and hence, efficiently preventing forgetting. A tiny replay buffer, although not critical, can be also integrated into our approach if forgetting occurs in the shared module. We established a new state of the art on CL benchmark datasets.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 139–154 (2018)
2. Azadi, S., Pathak, D., Ebrahimi, S., Darrell, T.: Compositional gan: Learning image-conditional binary composition. International Journal of Computer Vision pp. 1–16 (2020)
3. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational learning theory. pp. 92–100. ACM (1998)
4. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain separation networks. In: Advances in neural information processing systems. pp. 343–351 (2016)
5. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 532–547 (2018)
6. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with A-GEM. In: International Conference on Learning Representations (2019)
7. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: Continual learning with tiny episodic memories. arXiv preprint arXiv:1902.10486 (2019)
8. Chaudhuri, K., Kakade, S.M., Livescu, K., Sridharan, K.: Multi-view clustering via canonical correlation analysis. In: Proceedings of the 26th annual international conference on machine learning. pp. 129–136. ACM (2009)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
10. Ebrahimi, S., Elhoseiny, M., Darrell, T., Rohrbach, M.: Uncertainty-guided continual learning with bayesian neural networks. In: International Conference on Learning Representations (2020)
11. Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A., Wierstra, D.: Pathnet: Evolution channels gradient descent in super neural networks. arXiv preprint arXiv:1701.08734 (2017)
12. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The Journal of Machine Learning Research **17**(1), 2096–2030 (2016)
13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
14. Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: CyCADA: Cycle-consistent adversarial domain adaptation. In: International Conference on Machine Learning. pp. 1989–1998 (2018)
15. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
16. Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. In: International Conference on Learning Representations (2018)
17. Kim, H., Mnih, A.: Disentangling by factorising. arXiv preprint arXiv:1802.05983 (2018)

18. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences p. 201611835 (2017)
19. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
20. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
21. Lopez-Paz, D., et al.: Gradient episodic memory for continual learning. In: Advances in Neural Information Processing Systems. pp. 6467–6476 (2017)
22. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. arXiv preprint arXiv:1511.05644 (2015)
23. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
24. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989)
25. Nguyen, C.V., Li, Y., Bui, T.D., Turner, R.E.: Variational continual learning. In: ICLR (2018)
26. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR (2017)
27. Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., , Tesauro, G.: Learning to learn without forgetting by maximizing transfer and minimizing interference. In: International Conference on Learning Representations (2019)
28. Robins, A.: Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science **7**(2), 123–146 (1995)
29. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
30. Salzmann, M., Ek, C.H., Urtasun, R., Darrell, T.: Factorized orthogonal latent spaces. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 701–708 (2010)
31. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4548–4557. PMLR (2018)
32. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: Advances in Neural Information Processing Systems. pp. 2990–2999 (2017)
33. Shon, A., Grochow, K., Hertzmann, A., Rao, R.P.: Learning shared latent structure for image synthesis and robotic imitation. In: Advances in neural information processing systems. pp. 1233–1240 (2006)
34. Sinha, S., Ebrahimi, S., Darrell, T.: Variational adversarial active learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5972–5981 (2019)
35. Srivastava, R.K., Masci, J., Kazerounian, S., Gomez, F., Schmidhuber, J.: Compete to compute. In: Advances in neural information processing systems. pp. 2310–2318 (2013)

36. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7167–7176 (2017)
37. Van Der Maaten, L.: Accelerating t-sne using tree-based algorithms. The Journal of Machine Learning Research **15**(1), 3221–3245 (2014)
38. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in neural information processing systems. pp. 3630–3638 (2016)
39. Vitter, J.S.: Random sampling with a reservoir. ACM Transactions on Mathematical Software (TOMS) **11**(1), 37–57 (1985)
40. Xu, C., Tao, D., Xu, C.: A survey on multi-view learning. arXiv preprint arXiv:1304.5634 (2013)
41. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. In: International Conference on Learning Representations (2018)
42. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 3987–3995. PMLR (2017)
43. Zhang, M., Wang, T., Lim, J.H., Feng, J.: Prototype reminding for continual learning. arXiv preprint arXiv:1905.09447 (2019)