Adversarial Continual Learning

(Supplementary Materials)

Sayna Ebrahimi[1,2], Franziska Meier[1], Roberto Calandra[1],
Trevor Darrell[2], Marcus Rohrbach[1]
Facebook AI Research, Menlo Park, USA
UC Berkeley EECS, Berkeley, CA, USA

9 ACL Algorithm

Algorithm 1 shows the ACL pseudocode.

Algorithm 1 Adversarial Continual Learning (ACL))

```
1:  function TRAIN($\theta_P, \theta_S, \theta_D, \mathcal{D}^{tr}, \mathcal{D}^{ts}, m$)
2:      Hyper-parameters: $\lambda_1, \lambda_2, \lambda_3, \alpha_S, \alpha_P, \alpha_D$
3:      $R \leftarrow 0 \in \mathbb{R}^{T\times T}$
4:      $\mathcal{M} \leftarrow \{\}$
5:      $f_\theta^k = f(\theta_S \oplus \theta_P)$
6:      for $k = 1$ to T do
7:          for $e = 1$ to epochs do
8:              Compute $\mathcal{L}_{\text{adv}}$ for $S$ using $(x,t) \in \mathcal{D}_k^{tr} \cup \mathcal{M}$
9:              Compute $\mathcal{L}_{\text{task}}$ using $(x,y) \in \mathcal{D}_k^{tr} \cup \mathcal{M}$
10:             Compute $\mathcal{L}_{\text{diff}}$ using $P^k$, $S$, and $x \in \mathcal{D}_k^{tr}$
11:             $\mathcal{L}_{\text{ACL}} = \lambda_1\mathcal{L}_{\text{adv}} + \lambda_2\mathcal{L}_{\text{task}} + \lambda_3\mathcal{L}_{\text{diff}}$
12:             $\theta'_S \leftarrow \theta_S - \alpha_S\nabla\mathcal{L}_{\text{ACL}}$
13:             $\theta'_{P_t} \leftarrow \theta_{P^k} - \alpha_{P^k}\nabla\mathcal{L}_{\text{ACL}}$
14:             Compute $\mathcal{L}_{\text{adv}}$ for $D$ using $(S(x),t)$ and $(\mathbf{z}' \sim \mathcal{N}(\mu, \sum), t = 0)$
15:             $\theta'_{\text{D}} \leftarrow \theta_D - \alpha_D\nabla\mathcal{L}_{\text{adv}}$
16:         end for
17:         $\mathcal{M} \leftarrow$ UPDATEMEMORY($\mathcal{D}_k^{tr}, \mathcal{M}, C, m$)
18:         Store $\theta_{P^k}$
19:         $f_\theta^k \leftarrow f(\theta'_S \oplus \theta'_P)$
20:         $R_{k,\{1\cdots k\}} \leftarrow$ EVAL ($f_\theta^k, \mathcal{D}^{ts}_{\{1\cdots k\}}$)
21:     end for
22: end function

function UPDATEMEMORY($\mathcal{D}_k^{tr}, \mathcal{M}, C, m$)
    $s \leftarrow \frac{m}{C}$ ▷ $s :=$ # of samples per class
    for $c = 1$ to $C$ do
        for $i = 1$ to $n$ do
            $(x_i^k, y_i^k, t_i^k) \sim \mathcal{D}_k^{tr}$
            $\mathcal{M} \leftarrow \mathcal{M} \cup (x^k, y^k, t^k)$
        end for
    end for
    return $\mathcal{M}$
end function
function EVAL($f_\theta^k, \mathcal{D}^{ts}_{\{1\cdots k\}}$)
    for $i = 1$ to $k$ do
        $R_{k,i} = \text{Accuracy}(f_\theta^k(x,t), y)\text{for}(x,y,t) \in \mathcal{D}_i^{ts}$
    end for
    return $R$
end function
```

10 Datasets

Table 5a shows a summary of the datasets utilized in our work. From left to right columns are given as: dataset name, total number of classes in the dataset, number of tasks, image size, number of training images per task, number of validation images per task, number of test images per task, and number of classes in each task. Statistic of 5-Split MNIST and 5-Datasets experiments are given in 5b and 5c, respectively. We did not use data augmentation for any dataset.

(a) Statistics of utilized datasets. Datasets are as follows: a) 5-Split MNIST [20], b) Permuted MNIST [35], c) 20-Split CIFAR100 [19], d) 20-Split miniImageNet [9], e) 5-Datasets

Dataset	#Classes	#Tasks	Input Size	#Train/Task	#Valid/Task	#Test/Task
a	10	5	$1 \times 28 \times 28$	10	see Tab. 5b	see Tab. 5b
b	10	10/20/30/40	$1 \times 28 \times 28$	$51,000$	$9,000$	$10,000$
c	100	20	$3 \times 32 \times 32$	$2,125$	375	500
d	100	20	$3 \times 84 \times 84$	$2,125$	375	500
e	5×10	5	$3 \times 32 \times 32$	see Tab. 5c	see Tab. 5c	see Tab. 5c

(b) Number of training, validation, and test samples per task for 5-Split MNIST

Task number	T = 1 (0,1)	T=2 (2,3)	T=3 (4,5)	T=4 (6,7)	T=5 (8,9)
# Training samples	10766	10276	9574	10356	10030
# Validation samples	1899	1813	1689	1827	1770
# Test samples	2115	2042	1874	1986	1983

(c) Statistics of utilized datasets in 5-Datasets. MNIST, notMNIST, and Fashion MNIST are padded with 0 to become 32×32 and have 3 channels.

Dataset	MNIST	notMNIST	Fashion MNIST	CIFAR10	SVHN
# Training samples	51,000	15,526	9,574	42,500	62,269
# Validation samples	9,000	2,739	1,689	7,500	10,988
# Test samples	10,000	459	1,874	10,000	26,032

11 Results on 5-Split MNIST

Table 6 shows results obtained for 5-Split MNIST experiment described in section 7, respectively.

Table 6: Class Incremental Learning on 5-Split 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 and (††) denotes results are reported by [10]; All results are averaged over 3 runs, the standard deviation is provided in parenthesis

Method	**ACC%**	**BWT%**	Arch (MB)	Replay Buffer (MB)
EWC †† [18]	95.78(0.35)	-4.20(0.21)	1.1	-
HAT †† [31]	99.59(0.01)	0.00(0.04)	1.1	-
UCB ‡ [10]	99.63(0.02)	0.00(0.00)	2.2	-
VCL *[25]	95.97(1.03)	-4.62(1.28)	1.1	-
GEM* [21]	94.34(0.82)	-2.01(0.05)	6.5	0.63
VCL-C * [25]	93.6(0.20)	-3.10(0.20)	1.7	0.63
ORD-FT	65.96(3.53)	-40.15(4.27)	1.1	-
ORD-JT**	99.88(0.02)	-	189.3	-
ACL-JT** (Ours)	99.89(0.01)	-	190.8	-
ACL (Ours)	**99.76(0.03)**	0.01(0.01)	1.6	-

12 Effect of memory size on ACL

Fig. 2 shows the effect of memory size on ACL and memory-dependent baselines when 1, 3, 5, and 13 images per class are used during training where in the left it illustrates the memory effect for ACL and memory-dependent baselines. We also show how memory affects the BWT in ACL in Fig. 2 (right) which follows the same pattern as we observed for ACC. Numbers used to plot this figure with their standard deviation are given in Table 2.

13 Intransigence Measure

Table 7 shows our results for Intransigence (I) measure introduced in [5] and computed for the k-th task as below:

$$I_k = a_k^* - a_{k,k} \tag{6}$$

where a_k^* is the accuracy on the k-th task using a reference model trained with all the seen datasets up to task k and $a_{k,k}$ denotes the accuracy on the k-th task when trained up to task k in an incremental manner. $I_k \in [-1, 1]$.

Table 7 shows I evaluated at the end of tasks sequence for ACL and the strongest baselines (HAT [31] and ER-RES) in 5-Split MNIST, Permuted MNIST, 20-Split CIFAR100, and 20-Split miniImageNet experiments. We found ACL

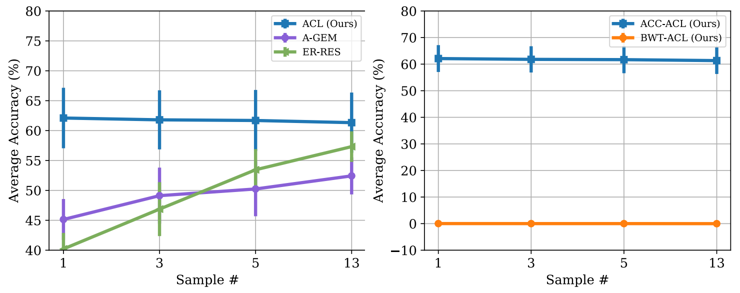

Fig. 2: *Left:* Comparing the replay buffer effect on ACC on 20-Split miniImageNet achieved by ACL against A-GEM [6] and ER-RES [7] when using 1, 3, 5, and 13 samples per classes within each task discussed in 5.2. *Right:* Insensitivity of ACC and BWT to replay buffer in ACL. Best viewed in color.

achieves the lowest I (lower the better) value compared to baselines, always negative, which means in ACL, learning tasks up to a specific task in hand increases the model's ability to learn new ones which is consistent with our idea of proposing a shared latent space for better initialization for new tasks.

Table 7: Intransigence measure results obtained for 5-Split MNIST, Permuted MNIST, 20-Split CIFAR100, and 20-Split miniImageNet experiments in HAT [31], ER-RES, and ACL. I_k denotes the measure computed at the end of the k-th task. A) 5-Datasets, B) Permuted MNIST (I_{10}), C) 20-Split CIFAR100 (I_{20}), D) 20-Split miniImageNet (I_{20})

	A	B	C	D
HAT [31]	-0.0001	-0.01	0.1	0.1
ER-RES	0.002	0.001	-0.002	-0.02
ACL (Ours)	**-0.5**	**-0.04**	**-0.03**	**-0.2**

14 Visualizing the effect of adversarial learning in ACL

Here we illustrate the role of adversarial learning in factorizing the latent space learned for continually learning a sequence of tasks using the t-distributed Stochastic Neighbor Embedding (t-SNE) [37] plots for the 20-Split miniImageNet experiment. The depicted results are obtained without using the orthogonality constraint, ($\mathcal{L}_{\text{diff}}$), to merely present the role of adversarial learning.

Fig. 3 visualizes the latent spaces of the shared and private modules trained with and without the discriminator. In the first row, we used the model trained on the entire sequence of 20-Split miniImageNet and evaluated it on the test-sets belonging to task #20 including 100 images for 5 classes, a total of 500 samples, which are color-coded with their class labels. In the second row, once again we used our final model trained on the entire sequence of 20-Split miniImageNet and tested it on the first 10 tasks of the sequence one at a time and plotted them all in a single figure for both shared and private modules with and without the discriminator where samples are color-coded with their task labels.

We first compare the discriminator's effect on the latent space generated by the shared modules (second column). The shared modules trained with adversarial loss, consistently appear as a uniformly mixed distribution of encoded samples both for task #20 and the first 10 tasks. In contrast, in the generated features without a discriminator in the fourth column for task #20, we observe a non-uniformly distributed mixture of features where small clusters can be found for some classes showing an entangled representation within each task. Similar to the pattern for task #20, we observe a mixed latent space representing the shared space for the first 10 tasks (fourth column).

The effect of the discriminator on the private modules' latent spaces is shown in the third and fifth columns where in the former, private modules trained with a discriminator, appear to be nearly successful in uncovering class labels (first row) and task labels (second row), although the final classification is yet to be happening in the private heads. As opposed to that, in the absence of the discriminator, private modules generate features as entangled as those generated by their shared module counterparts (first row). Private modules of the first 10 tasks is not as well-clustered as the private module trained with discriminator. As we can see in the fifth column, features for task #6 are spread into two parts with one close to task #7 and the other next to task #9.

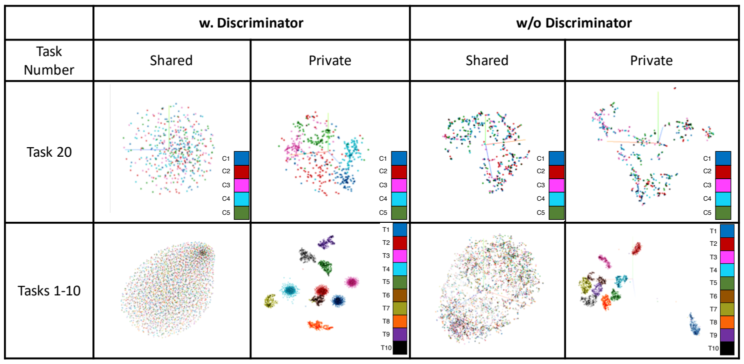

Fig. 3: Visualizing the effect of adversarial learning on the latent space generated by shared and private modules on 20-Split miniImageNet.