Unlearn to Learn Better: Continual Unlearning via Capacity Recycling

Pengxiang Wang[1] Shaobo Zhang[1] Hongbo Bo[2] Kedian Mu[1]
[1]Peking University, School of Mathematical Sciences, Beijing, China [2]University of Bristol, Bristol, UK
ECML PKDD 2026 Research Track
wangpengxiang@stu.pku.edu.cn

Continual Unlearning

Continual unlearning: Task-level unlearning from continual learning model
(An underexplored field combines continual learning + machine unlearning)
Objectives:
1. **Unlearning:** Minimal difference from retraining without unlearning tasks
2. **Learning:** Maximal performance on remaining tasks

Uncontrolled task removal causes catastrophic unlearning: damage to knowledge that should remain.

Problem Definition

We learn a sequence of tasks $T_1, T_2, \ldots$, where each task T_t has a distinct dataset D_t.
At the end of task T_t, an unlearning request may specify tasks $u(t) \subseteq \{T_1, \ldots, T_t\}$.

$$U(t) = \bigcup_{i=1}^{t} u(i), \qquad R(t) = \{T_1, \ldots, T_t\} \setminus U(t)$$

The resulting model should:
- Remove the knowledge of all accumulated unlearned tasks $U(t)$
- Match the retrained reference on the remaining tasks $R(t)$
- Maintain strong performance over the remaining tasks $R(t)$

In practice, each task may have an unlearnable age, so only tasks in the permitted set $P(t)$ require preparation for future unlearning.

Amnesiac Continual Learning (AmnesiacCL)

We propose **Amnesiac Continual Learning (AmnesiacCL)**, a general continual unlearning framework
- Can be integrated into any continual learning approach for unlearning
- Core mechanism: update deletion (simple but effective)

Continual Learning
Store task-wise update record

$$\theta_{l,ij}^{(t)} = \theta_{l,ij}^{(0)} + \sum_{\tau=1}^{t} \Delta\theta_{l,ij}^{(\tau)}$$

Continual Unlearning
Update deletion

$$\theta_{l,ij}^{(t-u(t))} = \theta_{l,ij}^{(t)} - \sum_{\tau \in u(t)} \Delta\theta_{l,ij}^{(\tau)}$$

Note: Reversing updates by substraction does not always perfectly invert the training process, and can lead to catastrophic unlearning. This is due to the stochastic and incremental nature of neural network training.

AmnesiacHAT

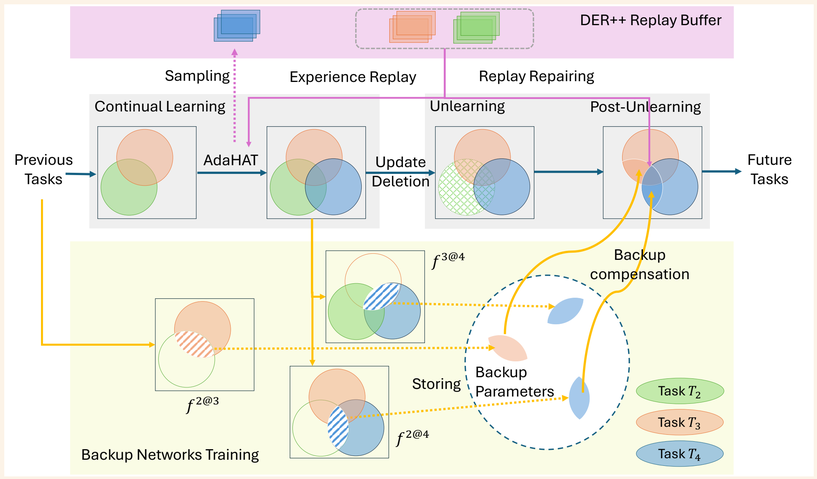
Architecture-based approaches are naturally compatible for continual unlearning and make it easier:
- Allocate task-specific subnetworks
- Less overlapping means less task interference
- Removing a task explicitly releases its allocated subspace
- Only overlapping parameters are affected by unlearning

AdaHAT ⇒ AmnesiacHAT

Continual Learning
- AdaHAT (Adaptive Hard Attention to the Task) architecture and training
- DER++ (Dark Experience Replay++) replay regularization

Continual Unlearning
- AmnesiacCL: task-wise parameter update storage and deletion

Post-Unlearning (to further mitigate catastrophic unlearning)
For only overlapping parameters:
- Backup compensation: immediate parameter replacement from backup networks
- Replay repairing: small-scale repair training to further restore performance



Backup Compensation and Replay Repairing

Post-Unlearning: Backup Compensation

Continual Learning
When learning task T_t, train parallel backup networks $f^{p@t}$ for all permitted unlearnable tasks $T_p \in P(t)$, assuming:
1. Non-overlapping parameters shared with the main network
2. Overlapping parameters freely updated from the state without T_p

$$h_l^{p@t} = f_l^{p@t}(h_{l-1}^{p@t}) m_l^t + f_l^t(h_{l-1}^{p@t})(1 - m_l^t)$$

Post-Unlearning
After unlearning T_p, replace parameters overlapping with all affected tasks T_t from backup networks after update deletion.

Post-Unlearning: Replay Repairing

A DER++ replay memory buffer is maintained during training to store a small amount of previous data, benefiting both continual learning and unlearning.

Continual Learning - Replay regularization

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + R_{\text{DER++}}(f(x^{\text{replay}}), z^{\text{replay}})$$

$$R_{\text{DER++}} = \lambda_{\text{ce}} \mathcal{L}_{\text{cls}} + \lambda_{\text{dist}} \| f(x) - z \|_2^2$$

Label replay and distillation from previous data

Post-Unlearning - Replay repairing

$$\mathcal{L}_{\text{repair}} = R_{\text{DER++}}(f(x^{\text{replay}}), z^{\text{replay}})$$

Repair for s steps, where replay data are from affected tasks by unlearning only

Discussion on Cost

AmnesiacHAT introduces mechanisms that increase computational and memory costs; however, none of them approaches the cost of full task-level computation.
- Updates are effectively restricted to subnetwork parameters; updates for update deletion can be stored sparsely

$$\Delta_s \theta^{T_p} = \Delta\theta^{T_p} \cdot m_\theta^{T_p}, \quad T_p \in P(t)$$

- We choose the underlying architecture-based framework AdaHAT, which reduces task interference and thus lowers the cost of the two post-unlearning processes. Only overlapping parameters are affected by unlearning and require backup and repair.
- Only need to prepare for permitted unlearnable tasks $P(t)$. Once a task exceeds its unlearnable age, its updates, backups, and replay data can be released.

Experiment Setup and Evaluation Metrics

Settings:
- Datasets: Permuted MNIST, Rotated MNIST, Split CIFAR-10, Combined-5
- Scenarios: Task-Incremental Learning (TIL), Domain-Incremental Learning (DIL)
- Task sequences:
  - 5 tasks (unlearn T_1 after T_2, T_2 after T_4, and T_5 after T_5)
  - 20 tasks (Permuted MNIST only)

Continual unlearning is a new research area. **We propose three primary metrics:**
- **Average Distribution Distance (ADD):** measures unlearning effectiveness by representation distance from the retrained reference model on unlearned tasks
- **Unlearning Accuracy Loss (UAL):** measures catastrophic unlearning by accuracy loss on remaining tasks relative to retrained reference model
- **Average Accuracy on Remaining Tasks (AAR):** measures continual learning performance on remaining tasks after unlearning

Main Results (excerpt)

Baseline	Permuted MNIST, 5 tasks, DIL			Permuted MNIST, 5 tasks, TIL		
	ADD↓	UAL↓	AAR↑	ADD↓	UAL↓	AAR↑
AmnesiacHAT	$45.87_{\pm 7.35}$	$\mathbf{5.32}_{\pm \mathbf{0.70}}$	$\mathbf{89.79}_{\pm \mathbf{0.68}}$	$\mathbf{41.30}_{\pm \mathbf{12.74}}$	$\mathbf{5.46}_{\pm \mathbf{0.50}}$	$\mathbf{89.54}_{\pm \mathbf{0.66}}$
DER++	$54.22_{\pm 2.21}$	$11.97_{\pm 2.29}$	$84.30_{\pm 2.09}$	$62.28_{\pm 3.14}$	$18.67_{\pm 4.28}$	$77.65_{\pm 3.93}$
DER	$58.42_{\pm 1.88}$	$18.50_{\pm 3.36}$	$77.57_{\pm 3.06}$	$61.62_{\pm 5.62}$	$28.94_{\pm 2.68}$	$67.14_{\pm 2.46}$
EWC	$\underline{\mathbf{42.12}_{\pm \mathbf{2.26}}}$	$17.10_{\pm 4.88}$	$75.62_{\pm 3.73}$	$56.77_{\pm 5.25}$	$20.87_{\pm 6.00}$	$65.19_{\pm 7.13}$
LwF	$45.39_{\pm 3.15}$	$43.53_{\pm 10.49}$	$47.35_{\pm 10.46}$	$50.06_{\pm 2.51}$	$27.94_{\pm 6.26}$	$59.96_{\pm 5.14}$
Finetuning	$43.19_{\pm 2.22}$	$15.08_{\pm 4.85}$	$77.63_{\pm 3.83}$	$54.71_{\pm 3.44}$	$14.91_{\pm 7.70}$	$71.67_{\pm 7.24}$
CLPU-DER++	$57.51_{\pm 2.06}$	$\underline{1.83_{\pm 0.41}}$	$\underline{94.44_{\pm 0.28}}$	$59.70_{\pm 2.52}$	$2.19_{\pm 0.31}$	$94.13_{\pm 0.32}$
Independent	–	–	–	$\underline{16.88_{\pm 1.45}}$	$\underline{0.05_{\pm 0.20}}$	$\underline{96.73_{\pm 0.13}}$

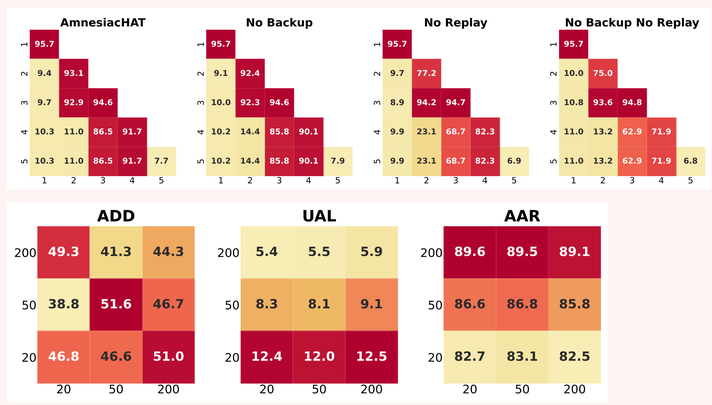
Baseline	Rotated MNIST, 5 tasks, DIL			Rotated MNIST, 5 tasks, TIL		
	ADD↓	UAL↓	AAR↑	ADD↓	UAL↓	AAR↑
AmnesiacHAT	$\underline{\mathbf{24.06}_{\pm \mathbf{3.33}}}$	$\mathbf{5.90}_{\pm \mathbf{1.42}}$	$\mathbf{89.10}_{\pm \mathbf{1.29}}$	$\mathbf{24.98}_{\pm \mathbf{2.50}}$	$\mathbf{6.90}_{\pm \mathbf{0.77}}$	$\mathbf{88.05}_{\pm \mathbf{0.69}}$
DER++	$38.50_{\pm 1.23}$	$37.95_{\pm 3.58}$	$56.58_{\pm 2.96}$	$34.00_{\pm 2.53}$	$39.52_{\pm 4.00}$	$56.39_{\pm 3.69}$
DER	$40.84_{\pm 0.71}$	$40.03_{\pm 3.57}$	$54.52_{\pm 2.90}$	$35.23_{\pm 2.79}$	$44.53_{\pm 5.29}$	$51.42_{\pm 4.79}$
EWC	$26.94_{\pm 3.50}$	$32.64_{\pm 6.39}$	$50.64_{\pm 5.88}$	$28.89_{\pm 3.82}$	$34.37_{\pm 5.80}$	$49.07_{\pm 5.54}$
LwF	$48.93_{\pm 5.16}$	$34.84_{\pm 7.02}$	$50.67_{\pm 6.36}$	$30.49_{\pm 3.79}$	$29.51_{\pm 5.62}$	$63.97_{\pm 4.83}$
Finetuning	$26.79_{\pm 2.38}$	$31.94_{\pm 6.42}$	$51.29_{\pm 5.71}$	$28.43_{\pm 1.35}$	$36.48_{\pm 4.05}$	$46.83_{\pm 2.53}$
CLPU-DER++	$61.67_{\pm 2.41}$	$\underline{-2.15_{\pm 0.52}}$	$\underline{96.68_{\pm 0.16}}$	$60.04_{\pm 2.17}$	$\underline{-0.81_{\pm 0.30}}$	$\underline{96.72_{\pm 0.07}}$
Independent	–	–	–	$\underline{13.19_{\pm 1.34}}$	$0.08_{\pm 0.22}$	$96.59_{\pm 0.16}$

Results:
- AmnesiacCL-based approaches show strong ADD
- AmnesiacHAT shows lowest UAL and highest AAR
- AmnesiacHAT approaches upper-bound but costly baselines (Independent, CLPU-DER++)

Conclusions: unlearning in AmnesiacCL is effective; AmnesiacHAT further mitigates catastrophic unlearning and maintains strong continual learning performance.

For complete experimental results, please refer to the paper.

Ablation and Hyperparameter Study



Backup compensation and replay repairing are both effective in mitigating catastrophic unlearning

Increasing replay buffer size M better mitigates catastrophic unlearning

Unlearning Benefit: Capacity Recycling

Continual learning algorithms seek a trade-off to get higher performance averaged on all tasks:
- **Stability:** preserve knowledge for previous tasks
- **Plasticity:** reserve representational space for new tasks

Update deletion generally benefits continual learning by recycling network capacity in a passive way:

Most CL approaches lack stability
Later tasks overwrite earlier tasks
⇓
Deleting later updates shortens the overwrite tail
⇓
Restore stability for earlier tasks

Architecture-based approaches lack plasticity
Task masks occupy a fixed parameter space
⇓
Deleting tasks releases their allocated subspaces
⇓
Restore plasticity for learning future tasks

AmnesiacCL compensates for whichever side of the stability-plasticity trade-off is lacking through capacity recycling, thus making continual learning better
- Metrics for stability: Backward Transfer (BWT)
- Metrics for plasticity: Forward Transfer (FWT)

The no-unlearning reference is the continual learning model trained on all tasks without unlearning.

$$SG = BWT_{R(N)} - BWT_{R(N)}^{\text{nounlearn}}, \qquad PG = FWT_{R(N)} - FWT_{R(N)}^{\text{nounlearn}}$$

Positive Stability Gain (SG) or Plasticity Gain (PG) shows which missing side is restored by unlearning.

Approach	$\text{BWT}_{R(N)}^{\text{nounlearn}}$	$\text{FWT}_{R(N)}^{\text{nounlearn}}$	SG	PG
Finetuning	$\underline{-48.86 \pm 2.81}$	-0.13 ± 0.08	$\mathbf{15.22 \pm 2.02}$	-6.17 ± 0.77
LwF	$\underline{-45.18 \pm 1.50}$	0.20 ± 0.10	$\mathbf{7.56 \pm 1.55}$	-9.83 ± 1.47
EWC	$\underline{-46.50 \pm 2.57}$	-0.08 ± 0.14	$\mathbf{10.02 \pm 4.54}$	-5.86 ± 1.48
DER	$\underline{-29.53 \pm 1.76}$	0.02 ± 0.10	$\mathbf{3.78 \pm 2.96}$	-6.19 ± 0.82
DER++	$\underline{-31.71 \pm 2.45}$	-0.00 ± 0.04	$\mathbf{18.26 \pm 2.32}$	-4.98 ± 1.78
AmnesiacHAT (original)	-16.28 ± 1.26	$\underline{-31.85 \pm 1.20}$	-1.31 ± 2.35	$\mathbf{11.13 \pm 3.20}$
AmnesiacHAT	-5.47 ± 0.70	-1.64 ± 0.09	-12.11 ± 1.94	-2.57 ± 0.64
CLPU-DER++	-31.71 ± 2.45	-0.00 ± 0.04	-1.37 ± 3.06	0.26 ± 0.09
Independent	0.00 ± 0.00	0.09 ± 0.10	0.00 ± 0.00	-0.09 ± 0.10

Results on Capacity Recycling Benefits

Conclusion

We studied continual unlearning as the joint problem of sequential learning and selective task removal.

We propose AmnesiacCL:
- A general continual unlearning framework that can be integrated into any continual learning approach
- A passive mechanism for capacity recycling that balances stability-plasticity trade-off and makes continual learning better

We develop AmnesiacHAT based on AmnesiacCL:
- Making unlearning easier and less catastrophic by reducing task interference
- Mitigates catastrophic unlearning through two post-unlearning processes

Insights: Unlearning is not only a trustworthy AI requirement; it can also passively make continual learning better.

Our work is implemented as a Python package for continual learning and unlearning research:
pengxiang-wang.com/projects/continual-learning-arena
(feel free to give it a star! :D)