Continual Learning and Unlearning Algorithms for Deep Neural Networks

Pengxiang Wang

Supervised by A.P. Kedian Mu

Peking University, School of Mathematical Sciences

2025-12-30

北京大学
PEKING UNIVERSITY

Table of contents I

Background: Continual Learning and Unlearning

AdaHAT: Adaptive Hard Attention to the Task in Task-Incremental Learning

FG-AdaHAT: Fine-Grained Neuron Importance Guided Architecture-Based Continual Learning

AmnesiacHAT: Continual Unlearning via Capacity Recycling in Architecture-Based Continual Learning

Continual Learning Arena (CLArena)

Summary

Background: Continual Learning and Unlearning

Continual Learning

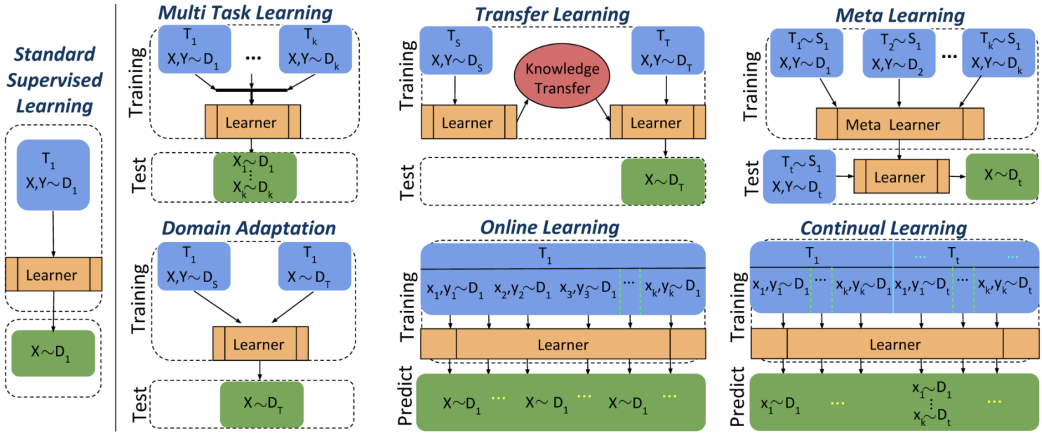
Continual Learning (CL), is a machine learning paradigm where an algorithm receives the data from tasks sequentially without the access to previous ones to learn a model that performs the best for all tasks.

(a.k.a. lifelong learning, incremental learning, sequential learning)

Key assumptions:

- Sequential tasks from different distributions
- Test for all tasks, but no access to previous task's data

Differences from Other Paradigms



Machine Unlearning

Machine unlearning, which becomes popular after 2020, deliberately removes the influence of specific data from a trained AI model.

Applications: data privacy, security, data quality, fairness, etc.

Retraining from scratch without the unlearning data is:

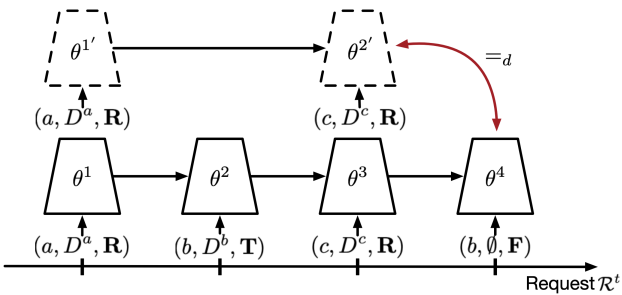
1. Computationally expensive
2. Not always feasible when the original data is unavailable

Unlearning is difficult.

Continual Unlearning

Continual unlearning: Unlearning specific tasks from a continual learning model

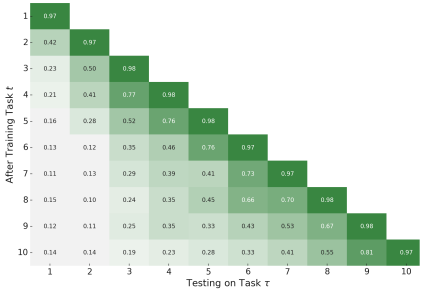
(A field that has been hardly explored before)

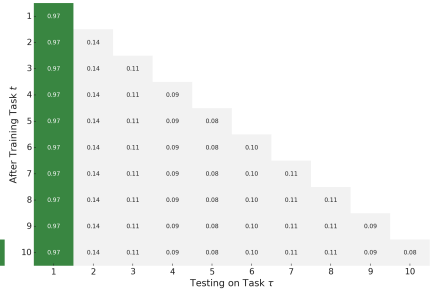


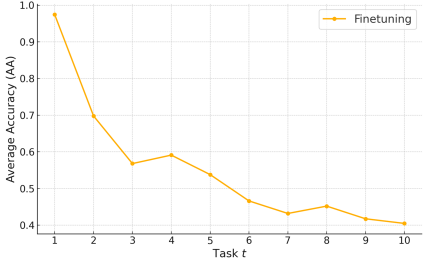
Objectives:

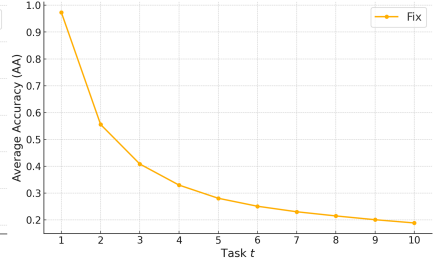
1. **Learning** objective: Maximal average performance on the remaining tasks
2. **Unlearning** objective: Minimal difference from retraining without the unlearning tasks

Baselines: Finetuning and Fix









Challenges

Catastrophic Forgetting and Unlearning

- Neural networks are holistic
  - Previous knowledge can hardly be preserved within neural networks after training different tasks
  - Unlearning causes catastrophic side effect
- The main problem that most CL and unlearning algorithms seek to address

Stability-Plasticity Dilemma

- The model cannot achieve both stability and plasticity at the same time
- Need to trade off the balance of stability-plasticity to get the best overall performance
- Unlearning also involves a similar dilemma between effective unlearning (plasticity) and less side effect on model (stability)

Challenges

Model Capacity and Resource Trade-Offs

Network Capacity Problem

- Any fixed-size network will eventually get full as infinite tasks arrive, leading to performance degradation on new tasks
- Expanding the network leads to memory overhead

Resource Trade-Offs Widely Exists

- To better prevent forgetting in CL:
  - More information about previous tasks generally needs to be stored
- To better unlearn in CUL:
  - More information about unlearning tasks generally needs to be stored
  - More mechanisms are involved before unlearning, introducing extra computation cost

Existing Approaches for Continual Learning

Replay-based Approaches

- Prevent forgetting by storing parts of the data from previous tasks
- Replay algorithms use them to consolidate previous knowledge
- E.g. iCaRL, GEM, DER, DGR ...

Regularization-based Approaches

- Add regularization terms constructed using information about previous tasks to the loss function when training new tasks
- E.g. LwF, EWC, SI, IMM, VCL, ...

Architecture-based Approaches

- Dedicate network parameters in different parts of the network to different tasks
- Keep the parameters for previous tasks from being significantly changed
- E.g. Progressive Networks, PackNet, DEN, Piggyback, HAT, CPG, UCL, ...

Existing Approaches for Continual Learning

Optimization-based Approaches

- Explicitly design and manipulate the optimization step
- For example, project the gradient not to interfere previous tasks
- E.g. GEM, A-GEM, OWM, OGD, GPM, RGO, TAG, ...

Representation-based Approaches

- Use special architecture or training procedure to create powerful representations
- Inspired from self-supervised learning, large-scale pre-training like LLMs
- E.g. Co2L, DualNet, prompt-based approaches (L2P, CODAPrompt, ...), CPT (continual pre-training)...

Very few works on continual unlearning so far. E.g. CLPU-DER++

My Work in Continual Learning

My PhD research mainly focuses on continual learning with:

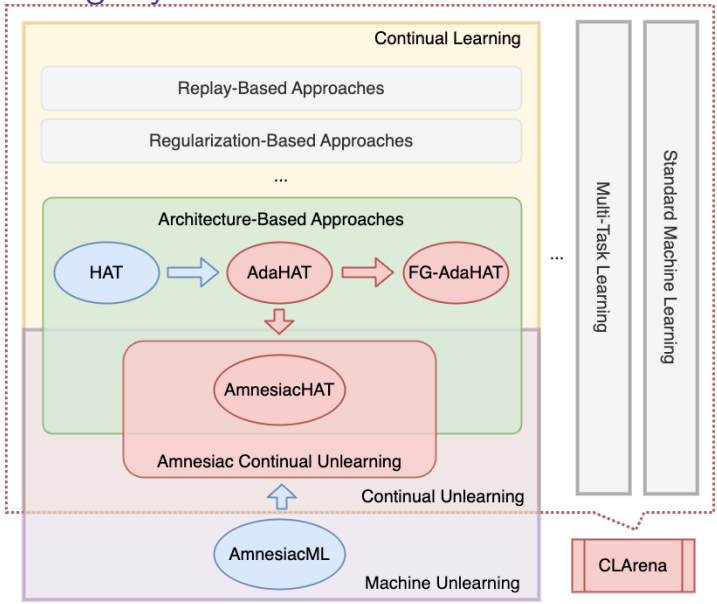
3 research papers:

1. **AdaHAT**: Adaptive Hard Attention to the Task in Task-Incremental Learning (published at ECML PKDD 2024, CCF-B)
2. **FG-AdaHAT**: Fine-Grained Neuron Importance Guided Architecture-Based Continual Learning (pending submission)
3. **AmnesiacHAT**: Continual Unlearning via Capacity Recycling in Architecture-Based Continual Learning (pending submission)

1 open-source software:

- **CLArena**: A Python package for continual learning research (published at PyPI)

Relationships Among My Work



AdaHAT: Adaptive Hard Attention to the Task in Task-Incremental Learning

Motivation for AdaHAT

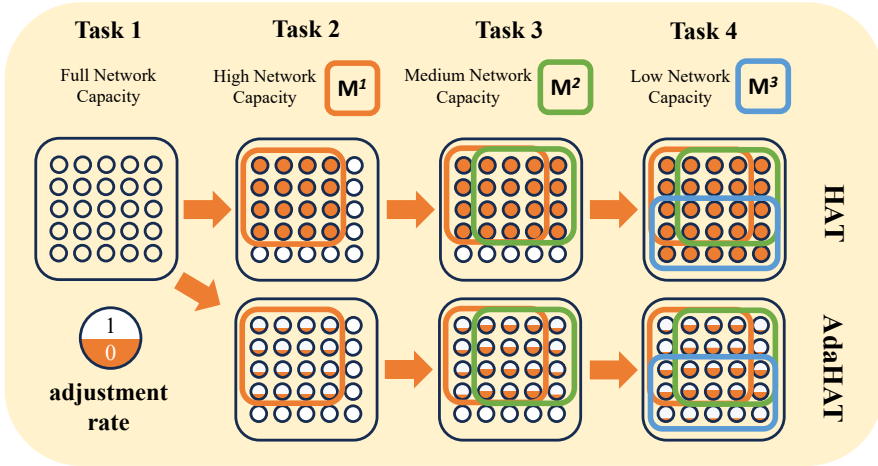
Architecture-based approaches (a.k.a. parameter isolation) belong to one of the main categories of continual learning algorithms:

- A distinctly different strategy that decomposes the network architecture
- Dedicate different parts of a neural network to different tasks

HAT (Hard Attention to the Task) is one of the most representative architecture-based approaches:

- Hard (binary) masks on layers
- Treat the masks as model parameters
- Masks condition on gradients directly

HAT: Hard Attention to the Task



Network Capacity Problem in HAT

HAT's hard gradient clipping mechanism allows no update for parameters masked by previous tasks:

$$g'_{l,ij} = a_{l,ij} \cdot g_{l,ij}, \ a_{l,ij} \in \{0, 1\}$$

More tasks come in

↓

More active parameters become static

↓

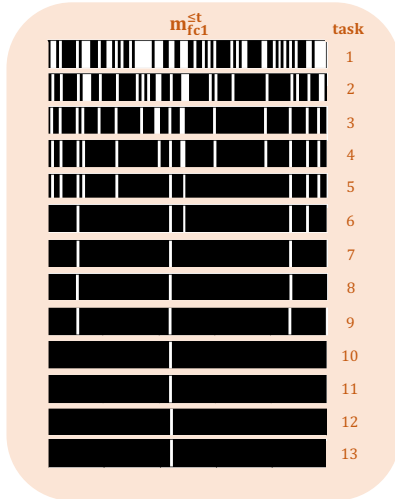
Insufficient network capacity

↓

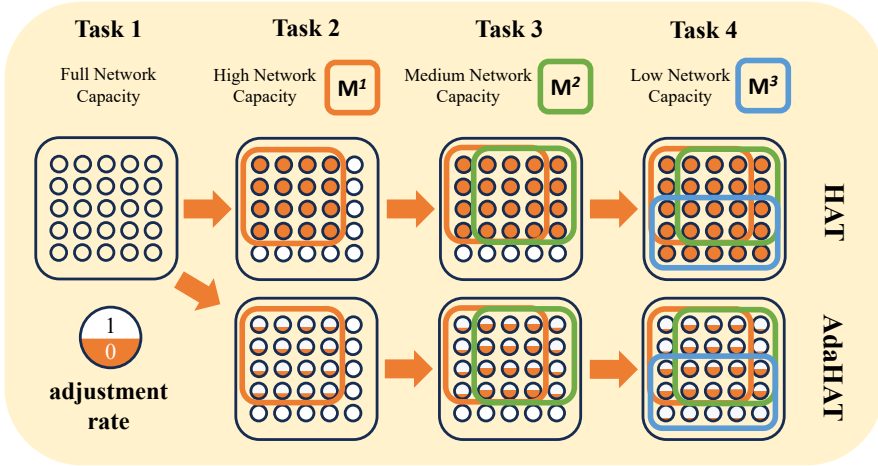
Significantly affect performance on new tasks

↓

Learning plasticity reduced, inbalanced stability-plasticity trade-off



AdaHAT: Adaptive Hard Attention to the Task



HAT $\Rightarrow$ **AdaHAT (Adaptive HAT)**

AdaHAT: Adaptive Hard Attention to the Task

AdaHAT **soft-clips gradients**, allowing minor updates for parameters masked by previous tasks:

$$g'_{l,ij} = a^{\star}_{l,ij} \cdot g_{l,ij}, \;\; a^{\star}_{l,ij} \in [0, 1]$$

The adjustment rate $a^{\star}_{l,ij}$ now is an adaptive controller, guided by two pieces of information about previous tasks:

- **Parameter Importance**: how many tasks the target parameter has been used for
- **Network Sparsity**: how many parameters are currently used for all previous tasks

AdaHAT: Adaptive Hard Attention to the Task

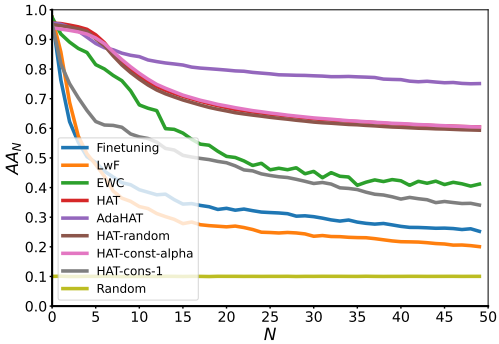
Adaptive Adjustment Rate (AdaHAT)

$$a^{\star}_{l,ij} = \frac{r_l}{\min\left(m^{<t,\mathsf{sum}}_{l,i}, m^{<t,\mathsf{sum}}_{l-1,j}\right) + r_l}, \;\; r_l = \frac{\alpha}{R\left(\mathsf{M}^t, \mathsf{M}^{<t}\right) + \epsilon}$$

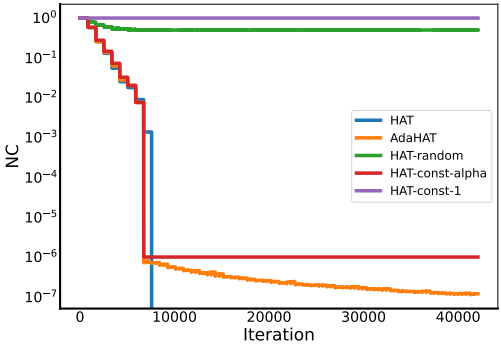
Adjustment Rate (HAT)

$$a_{l,ij} = 1 - \min\left(m^{<t}_{l,i}, m^{<t}_{l-1,j}\right)$$

Experiment Highlights



Results on long task sequences



Network capacity usage

FG-AdaHAT: Fine-Grained Neuron Importance Guided Architecture-Based Continual Learning

Motivation for FG-AdaHAT

The adaptive gradient clipping mechanism in AdaHAT:

- Parameter Importance is an integer value from 0 to $t-1$
- Network Sparsity is a single scalar value for the whole network

Fine-grained task information plays a crucial role among other continual learning approaches, meaningfully contributing to guiding the training of new tasks.

AdaHAT $\Rightarrow$ **FG-AdaHAT (Fine-Grained AdaHAT)**

$$a_{l,ij}^{\text{FG-AdaHAT}} = \alpha \left[c^t \cdot \text{Agg}\left(I_{l,i}^{<t}, I_{l-1,j}^{<t} \right) + \alpha \right]^{-1}, c^t = (t + b_L) \cdot \left[R\left(\mathsf{M}^t, \mathsf{M}^{<t} \right) + b_R \right]$$

Fine-Grained Neuron Importance in FG-AdaHAT

Neuron Importance measures how important a neuron is to previous tasks, which is finer-grained than Parameter Importance and Network Sparsity.

$$I_{l,i}^{<t} = \sum_{\tau<t} I_{l,i}^{\tau},\ I_{l,i}^{\tau} = \frac{1}{|D^{\tau}|} \sum_{(\mathbf{x},y)\in D^{\tau}} I_{l,i}^{\tau}(\mathbf{x},y)$$

It can be constructed from training information:

- Input Gradients (IG): $I_{l,i}^{\tau}(\mathbf{x},y) = \sum_j |g'_{l,ij}|$
- Output Gradients (OG): $I_{l,i}^{\tau}(\mathbf{x},y) = \sum_i |g'_{l+1,ij}|$
- Contribution Utility (CU): $I_{l,i}^{\tau}(\mathbf{x},y) = |h_{l,j}(\mathbf{x})| \sum_i |w_{l+1,ij}|$
- Input Contribution Utility (ICU): $I_{l,i}^{\tau}(\mathbf{x},y) = |h_{l,i}(\mathbf{x})| \sum_j |w_{l,ij}|$
- Activation Fisher Information (AFI): $I_{l,i}^{\tau}(\mathbf{x},y) = |h_{l,i}(\mathbf{x})| \sum_j |g'_{l,ij}|^2$

Fine-Grained Neuron Importance in FG-AdaHAT

Layer attribution methods from **Explainable AI (XAI)** provides more fine-grained neuron importance measures:

- Feature Ablation (FA): perturbation-based attribution
- Internal Influence (II): gradient-based attribution
- Gradient SHAP (GS): gradient-based attribution, using Shapley values
- DeepLIFT (DL): backpropagation-based attribution

Experiment Highlights

Approach	Permuted MNIST		Split CIFAR-100		Combined-20	
	AA (↑)	FR (↑)	AA (↑)	FR (↑)	AA (↑)	FR (↑)
Finetuning	32.06 ± 0.96	−73.19 ± 1.11	34.19 ± 1.84	−72.47 ± 3.34	10.88 ± 0.44	−74.45 ± 1.72
LwF	32.95 ± 1.35	−72.12 ± 1.55	31.78 ± 2.36	−77.20 ± 5.34	10.92 ± 0.91	−74.36 ± 1.04
HAT	67.10 ± 0.56	−30.68 ± 0.65	40.12 ± 2.17	−59.12 ± 4.01	17.05 ± 2.57	−74.63 ± 4.16
AdaHAT	79.91 ± 1.47	−12.43 ± 6.46	44.93 ± 1.29	−50.28 ± 2.51	16.46 ± 2.49	−68.31 ± 4.74
FG-AdaHAT (IG)	84.05 ± 1.71	−10.12 ± 1.86	45.41 ± 3.30	−49.36 ± 5.87	**24.67 ± 4.15**	−56.65 ± 4.74
FG-AdaHAT (OG)	**85.05 ± 0.71**	**−8.92 ± 0.82**	45.54 ± 2.19	−49.55 ± 3.69	22.05 ± 2.77	−59.11 ± 4.17
FG-AdaHAT (CU)	84.11 ± 0.91	−10.05 ± 1.05	45.71 ± 3.47	−49.12 ± 6.42	22.26 ± 1.69	−59.98 ± 3.76
FG-AdaHAT (ICU)	84.97 ± 0.63	−9.01 ± 0.72	**49.66 ± 1.88**	**−40.93 ± 3.75**	19.95 ± 3.93	−62.37 ± 4.80
FG-AdaHAT (AFI)	83.43 ± 1.28	−10.88 ± 1.47	47.00 ± 1.79	−46.66 ± 3.31	20.09 ± 5.16	−61.29 ± 5.90
FG-AdaHAT (FA)	84.86 ± 0.89	−9.14 ± 1.03	–	–	–	–
FG-AdaHAT (II)	83.66 ± 0.74	−10.60 ± 0.85	45.70 ± 1.66	−48.85 ± 2.85	22.32 ± 1.01	−59.10 ± 4.24
FG-AdaHAT (GS)	83.91 ± 0.70	−10.30 ± 0.81	46.43 ± 3.83	−47.40 ± 7.32	24.25 ± 6.52	**−56.05 ± 9.23**
FG-AdaHAT (DL)	84.70 ± 0.57	−9.34 ± 0.66	47.70 ± 4.64	−45.02 ± 7.92	23.71 ± 4.49	−57.79 ± 5.34

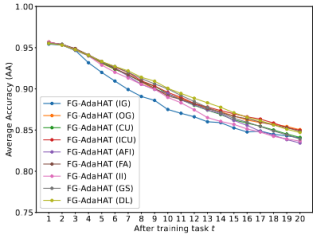
Performance on 3 continual learning benchmarks

Experiment Highlights

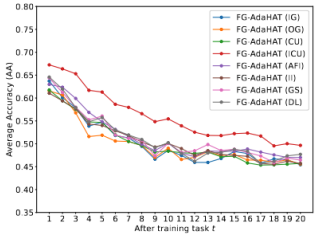
Approach	Permuted MNIST		Split CIFAR-100		Combined-20	
	BWT (↑)	FWT (↑)	BWT (↑)	FWT (↑)	BWT (↑)	FWT (↑)
Finetuning	−68.52 ± 0.96	0.42 ± 0.02	−51.17 ± 1.87	10.70 ± 0.40	−64.88 ± 0.57	9.87 ± 0.22
LwF	−67.42 ± 1.34	0.25 ± 0.05	−51.78 ± 3.40	8.88 ± 1.06	−64.75 ± 0.81	9.70 ± 0.21
HAT	−0.00 ± 0.00	−31.12 ± 0.57	−21.40 ± 2.40	−12.43 ± 0.72	−19.18 ± 3.98	−27.31 ± 4.40
AdaHAT	−12.46 ± 1.42	−5.19 ± 0.09	−27.23 ± 1.06	−1.60 ± 0.97	−40.91 ± 2.29	−6.16 ± 1.29
FG-AdaHAT (IG)	−7.73 ± 1.67	−5.55 ± 0.17	−19.90 ± 3.13	−8.13 ± 0.82	−33.90 ± 2.45	−5.11 ± 2.62
FG-AdaHAT (OG)	−5.33 ± 0.70	−6.90 ± 0.19	−19.66 ± 2.08	−8.12 ± 1.01	−34.55 ± 2.13	−6.56 ± 1.80
FG-AdaHAT (CU)	−6.09 ± 0.90	−7.14 ± 0.11	−22.61 ± 2.62	−5.00 ± 0.77	−35.31 ± 1.57	−5.79 ± 0.71
FG-AdaHAT (ICU)	−5.57 ± 0.67	−6.75 ± 0.11	−23.60 ± 1.70	−0.41 ± 0.69	−38.10 ± 2.72	−5.37 ± 1.55
FG-AdaHAT (AFI)	−7.32 ± 1.24	−6.62 ± 0.14	−21.92 ± 2.23	−4.40 ± 1.18	−37.06 ± 2.86	−6.58 ± 2.71
FG-AdaHAT (FA)	−6.71 ± 0.87	−5.73 ± 0.09	—	—	—	—
FG-AdaHAT (II)	−6.02 ± 0.70	−7.69 ± 0.12	−20.57 ± 1.54	−7.01 ± 1.58	−34.48 ± 4.14	−6.59 ± 3.23
FG-AdaHAT (GS)	−4.75 ± 0.69	−8.68 ± 0.25	−20.92 ± 2.61	−6.06 ± 1.83	−34.58 ± 4.19	−4.83 ± 1.80
FG-AdaHAT (DL)	−3.89 ± 0.57	−8.70 ± 0.25	−20.03 ± 3.92	−5.63 ± 1.11	−34.65 ± 2.26	−5.29 ± 1.70

Stability-plasticity trade-off on 3 continual learning benchmarks

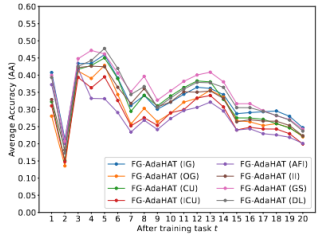
Experiment Highlights



(a) Permuted MNIST



(b) Split CIFAR-100

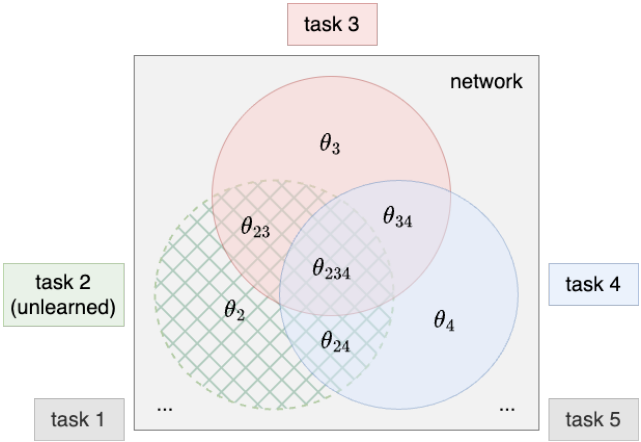


(c) Combined-20

Comparison of 9 fine-grained neuron importance on 3 benchmarks

AmnesiacHAT: Continual Unlearning via Capacity Recycling in Architecture-Based Continual Learning

Motivation for AmnesiacHAT



Architecture-based approaches are naturally compatible for continual unlearning:

- Less overlapping, less task interference
- Only overlapping parameters are affected by unlearning
- We choose our fix-sized architecture-based approach AdaHAT

AdaHAT $\Rightarrow$ **AmnesiacHAT**

AmnesiacHAT Algorithm

Continual Learning

- **AdaHAT** architecture and training process
- **DER** (Dark Experience Replay) replay regularization

Continual Unlearning

- **Amnesiac task-wise update**: task-wise parameter update storage and deletion

Post-unlearning processing for overlapping parameters:

- **Backup compensation**: immediate parameter replacement from backup networks
- **Replay repairing**: small-scale repair training to further restore performance

Unlearning Algorithm: Amnesiac Task-Wise Update

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

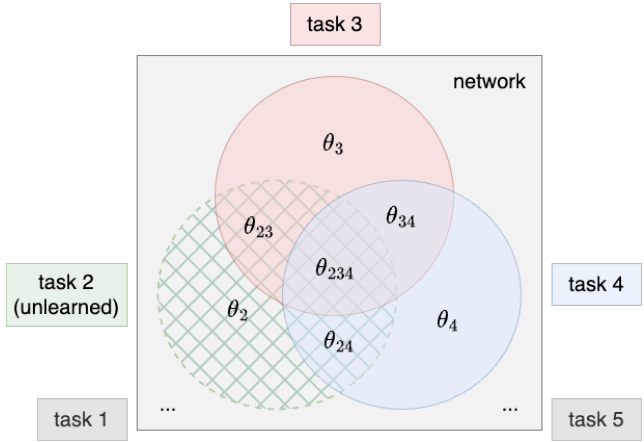
$$\theta_{l,ij}^{(t-u(t))} = \theta_{l,ij}^{(t)} - \sum_{\tau \in u(t)} \Delta\theta_{l,ij}^{(\tau)}$$

Task-wise update record can be stored ***sparsely*** (task-owned & residual) to save memory:

$$\Delta_s\theta_{l,ij}^{(\tau)} = \Delta\theta_{l,ij}^{(\tau)} \cdot m_{l,i}^{\tau},\ \Delta_r\theta_{l,ij}^{(\tau)} = \Delta_r\theta_{l,ij}^{(\tau-1)} + \Delta\theta_{l,ij}^{(\tau)} \cdot (1 - m_{l,i}^{\tau})$$

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

Unlearning Algorithm: Amnesiac Task-Wise Update



To unlearn task 2, delete $\Delta\theta^{(2)}$

Unlearning Algorithm: Backup Compensation

Continual Learning

When learning task t, train parallel **backup networks** $B^{t_u \to t}$ for all unlearnable tasks t_u, assuming:

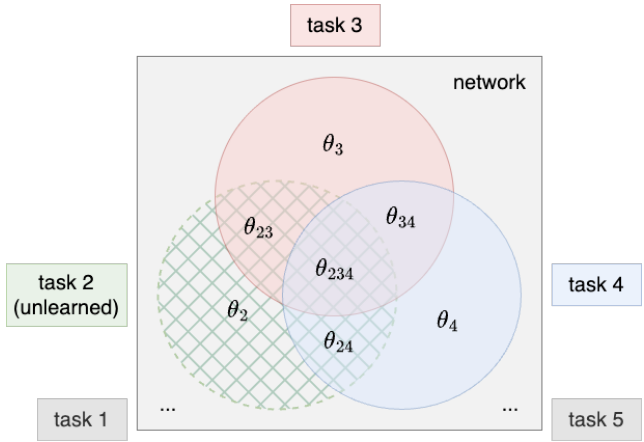
1. Non-overlapping parameters shared with the main network
2. Overlapping parameters freely updated from scratch

$$\mathbf{h}_l^{t_u \to t} = B_l^{t_u \to t}(\mathbf{h}_{l-1}^{t_u \to t})\mathbf{m}_l^t + B_l(\mathbf{h}_{l-1}^{t_u \to t})(1 - \mathbf{m}_l^t)$$

Continual Unlearning

When unlearning t_u, replace parameters overlapping with all affected tasks t from backup networks after update deletion.

Unlearning Algorithm: Backup Compensation



Backup when training task 3, 4: $B^{2\to 3}, B^{2\to 4}$

Compensation after unlearning task 2: $\theta_{23}, \theta_{24}, \theta_{234}$

Unlearning Algorithm: 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

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

$$R_{\text{DER}} = \|f(\mathbf{x}) - \mathbf{z}\|_2^2$$

Knowledge distillation from previous data

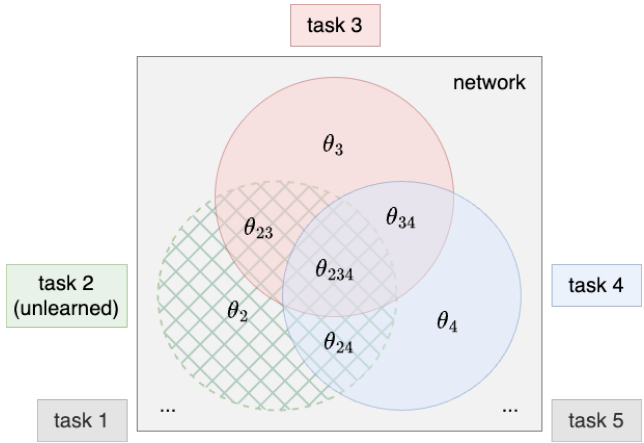
Continual Unlearning

Replay repairing

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

Repair for s_{repair} steps, where replay data are from affected tasks by unlearning only

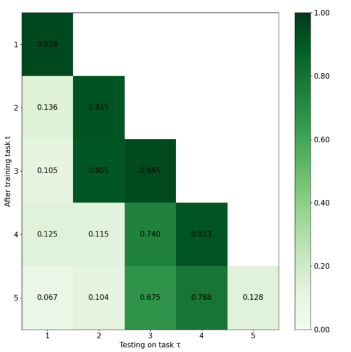
Unlearning Algorithm: Replay Repairing

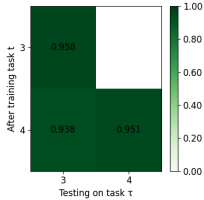


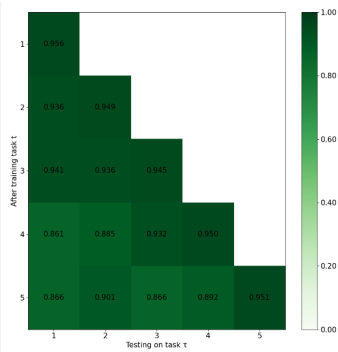
Repair after unlearning task 2: $\theta_{23}, \theta_{24}, \theta_{234}$

Using replay data from task 3, 4

Experiment Highlights



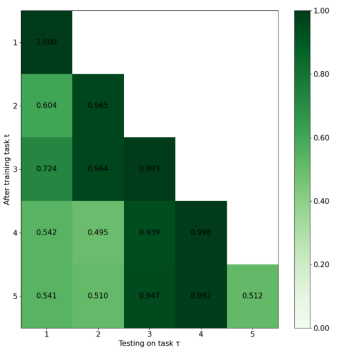


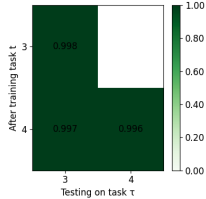


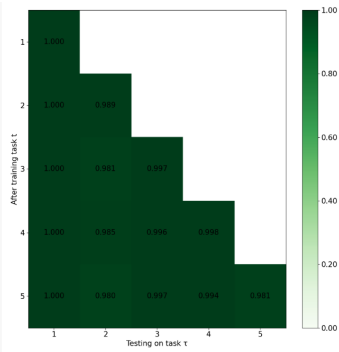
Original, reference retrain, reference original experiments on Permuted MNIST, 5 tasks

Unlearning requests: unlearn task 1 after 2, 2 after 4, 5 after 5

Experiment Highlights



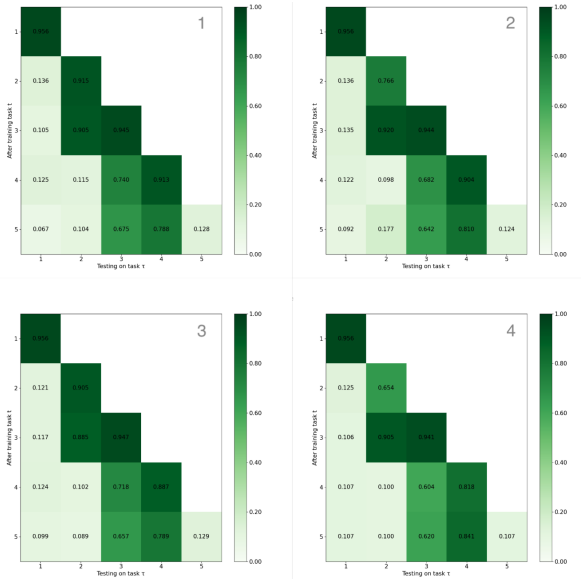




Original, reference retrain, reference original experiments on Split MNIST, 5 tasks

Unlearning requests: unlearn task 1 after 2, 2 after 4, 5 after 5

Experiment Highlights

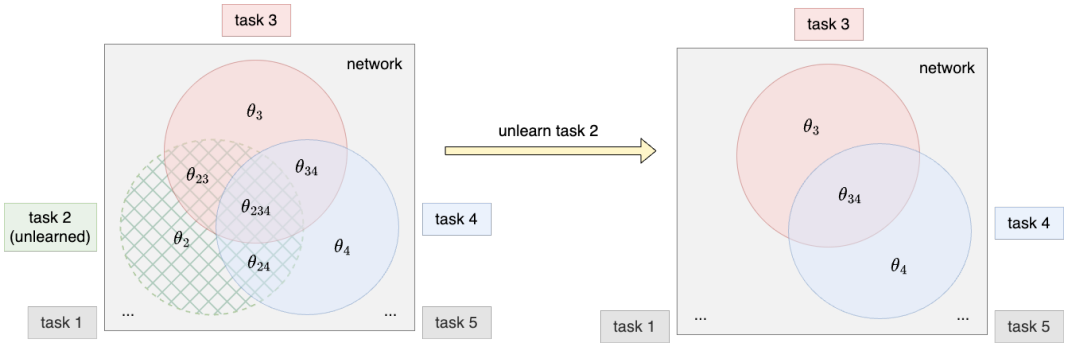


Ablation study on
Permuted MNIST, 5 tasks

1. AmnesiacHAT
2. No backup compensation
3. No replay repairing
4. Neither no

Additional Benefit: Capacity Recycling

When a task is unlearned, the parameters that were dedicated to that task become free and can be reused for future tasks.

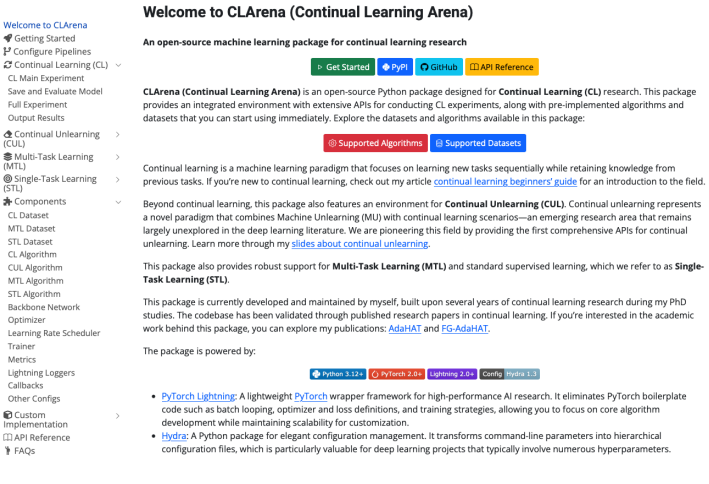


How to measure it? *Compare the performance without unlearning (AGR metric)*

Continual Learning Arena (CLArena)

Continual Learning Arena (CLArena)

Continual Learning Arena (CLArena) is an open-source Python package for continual learning research, developed and documented entirely by myself.



Key Features of CLArena

1. Provide pipelines for various machine learning paradigms:

- Continual Learning (CL)
- Continual Unlearning (CUL)
- Multi-Task Learning (MTL)
- Single-Task Learning (STL)

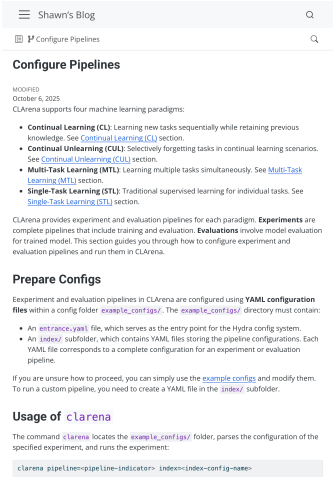
2. Provide implementations of:

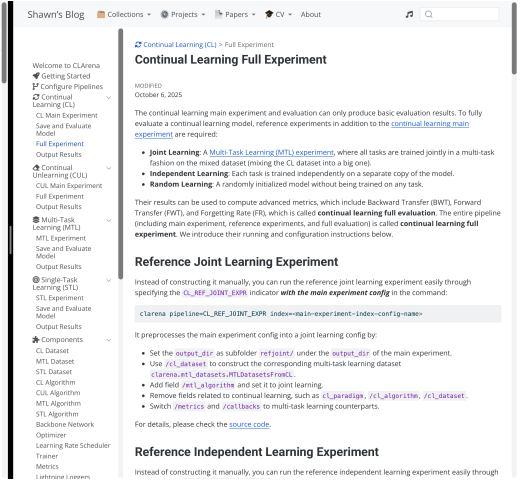
- Mainstream CL algorithms: LwF, EWC, HAT, WSN, CBP, etc.
- Common CL / MTL / STL benchmarks: permuted, split, combined, etc.
- Neural network architectures: MLP, ResNet, HAT-based models, etc.

3. Provide framework for building custom datasets, models, algorithms, metrics, callbacks, etc.

4. Full code support for my 3 works above, ensuring reproducibility

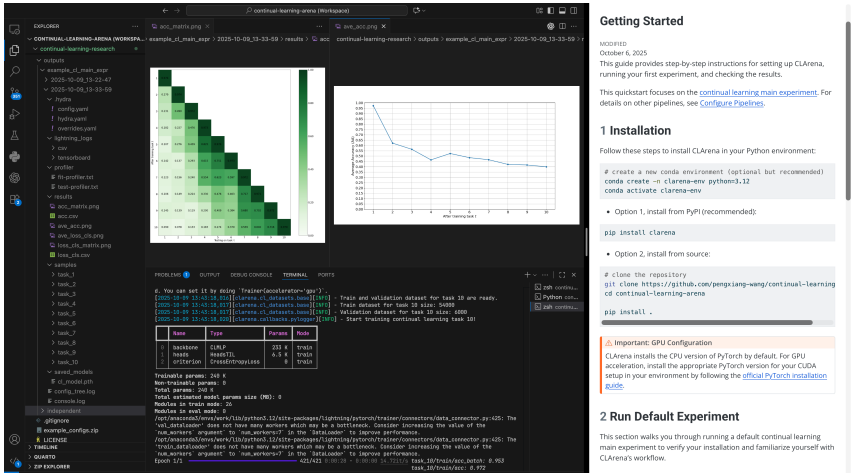
Ease-of-Use Features





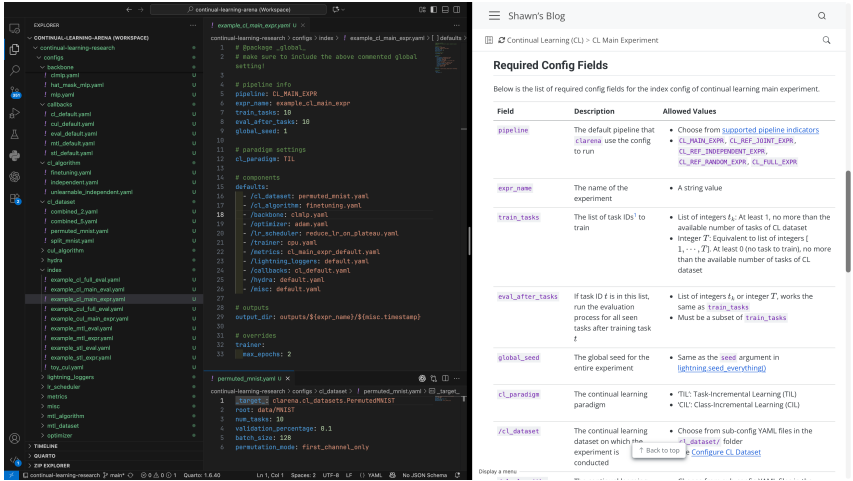
Well-structured and detailed documentation

Ease-of-Use Features



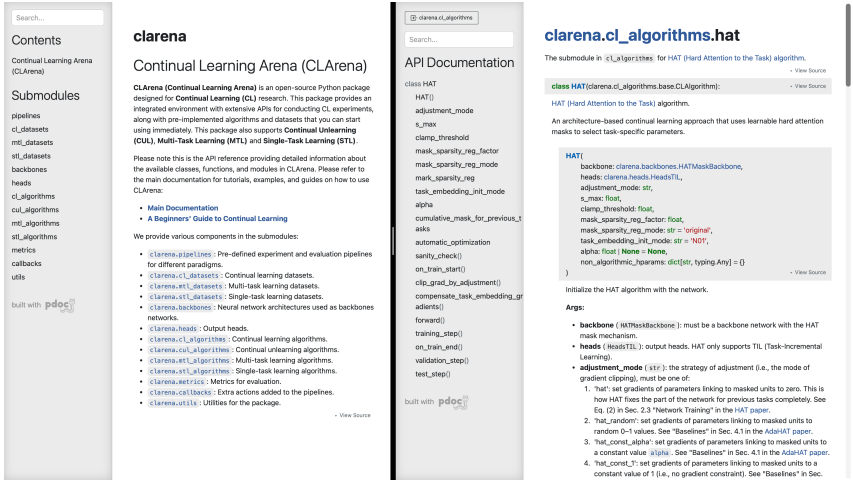
Get started quickly with tutorials and examples

Ease-of-Use Features



An easy configuration system based on YAML, each config field clearly explained

Ease-of-Use Features



Complete API reference for custom implementation in the framework

Package Information

Main Page / Documentation

pengxiang-wang.com/projects/continual-learning-arena

GitHub (feel free to give it a star! :D)

github.com/pengxiang-wang/continual-learning-arena

PyPI (‘pip install clarena‘)

pypi.org/project/clarena

API Reference

pengxiang-wang.com/projects/continual-learning-arena/docs/api-reference

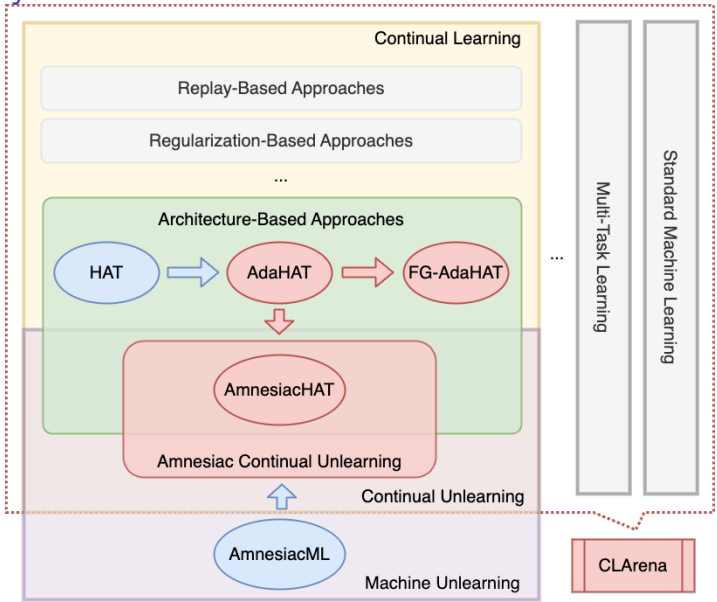
Summary

Summary of My PhD Work

Overall, my PhD work dives deep into continual learning through architecture-based approaches, across comprehensive contributions to this area in both width and depth:

- Novel well-performed continual learning approaches: **AdaHAT**, **FG-AdaHAT** variants
- Novel algorithm framework: **FG-AdaHAT**
- Paradigm problem hardly explored before: **continual unlearning**, and its solution: **AmnesiacHAT**
- Novel software platform for continual learning research: **CLArena**

Summary of My PhD Work



Solved Problems of My Work

AdaHAT / FG-AdaHAT:

- Balances the stability-plasticity trade-off in continual learning
- Alleviates the network capacity problem that architecture-based approaches generally suffer from
- Addresses the challenges of continual learning on long task sequences

AmnesiacHAT:

- Provides a solution to continual unlearning problem: to erase knowledge learned from specific tasks without affecting other learned tasks
- Addresses the network capacity problem from a new perspective: capacity recycling through unlearning

Thank You

Thank you for your attention!

My website: pengxiang-wang.com

My email: wangpengxiang@stu.pku.edu.cn