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

ECML PKDD 2024 Research Track

Pengxiang Wang* Hongbo Bo[2,3] Jun Hong[4] Weiru Liu[3] Kedian Mu[1]

[1] Peking University, School of Mathematical Sciences, Beijing, China

[2] Newcastle University, Population Health Sciences Institute, Newcastle, UK

[3] University of Bristol, School of Engineering Mathematics and Technology, Bristol, UK

[4] University of the West of England, School of Computing and Creative Technologies, Bristol, UK

Continual Learning and Catastrophic Forgetting

Continual Learning

- A machine learning paradigm
- Learn sequential tasks and adapt to new information over time
- One of the key features of human intelligence

Catastrophic Forgetting

- Drastic performance degradation on previous tasks after learning new tasks
- A major problem for continual learning algorithm to address

Problem Definition

Continual Learning (CL): learning a sequence of tasks $t = 1, \cdots, N$ in order, with datasets $D^t = \{x^t, y^t\}$

Task-Incremental Learning (TIL): continual learning scenario, aim to train a model f that performs well on all learned tasks

$$\max_f \sum_{t=1}^{N} \text{metric}(f(x^t), y^t), \{x^t, y^t\} \in D^t$$

Key assumptions when training and testing task t:

- No access to the entire data from previous tasks $1, \cdots, t-1$
- Testing on all seen tasks $1, \cdots, t$
- For TIL testing, task ID t of each test sample is known by the model

Existing Approaches for TIL

Replay-based Approaches

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

Regularization-based Approaches

- Introduce 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

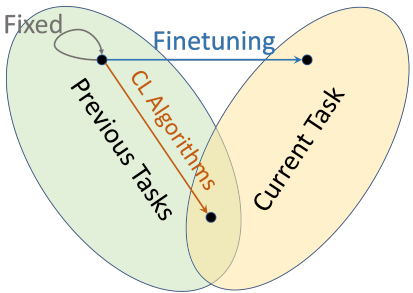
- Allocate network parameters in different parts of the network to different tasks
- Focus on reducing representational overlap in the network
- Keep the parameters learned in previous tasks from being significantly changed
- E.g. Progressive Networks, PathNet, PackNet, Piggyback, HAT, CPG, SupSup, ...

Stability-Plasticity Trade-Off

Continual learning is a trade-off between stability and plasticity.

- **Stability:** preserve knowledge for previous tasks
- **Plasticity:** reserve representational space for new tasks

Continual learning algorithms must trade them off to get higher performance averaged on all tasks.



For replay-, regularization-based approaches:

- Inject stability in their forgetting prevention mechanisms
- But generally still lean towards plasticity

For architecture-based approaches:

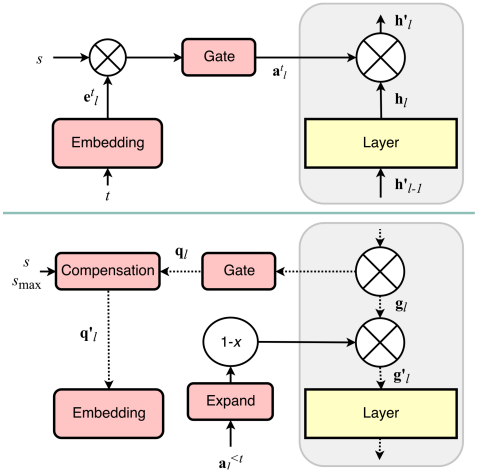
- Distinctly different strategies that overemphasize stability
- Shifting the trade-off towards stability instead

HAT: Hard Attention to the Task

HAT (Hard Attention to the Task) is one of the most representative architecture-based approaches. Our work AdaHAT provides an extension to HAT.

Key features:

- Hard (binary) attention vectors (masks) on layers, allocating the part of the network to each task
- Treat the masks as model parameters, which means masks are learned
- Masks condition on gradients directly; masked parameters won't be updated



Mechanism Details of HAT

Layer-wise attention vectors (masks) are learned to pay hard (binary) attention on units in each layer $l = 1, \cdots, L-1$ to a new task t:

$$\mathbf{m}_l^{\leq t} = \max\left(\mathbf{m}_l^t, \mathbf{m}_l^{\leq t-1}\right)$$

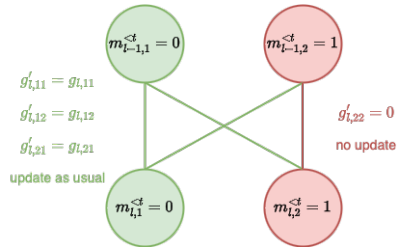
Binary values are gated from real-value task embeddings which is learnable:

$$\mathbf{m}_l^t = \sigma\left(s\mathbf{e}_l^t\right)$$

Masks **hard-clip gradients** of parameters:

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

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



Problem 1: Insufficient Network Capacity

HAT's hard-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 arrive

↓

More active parameters become static

↓

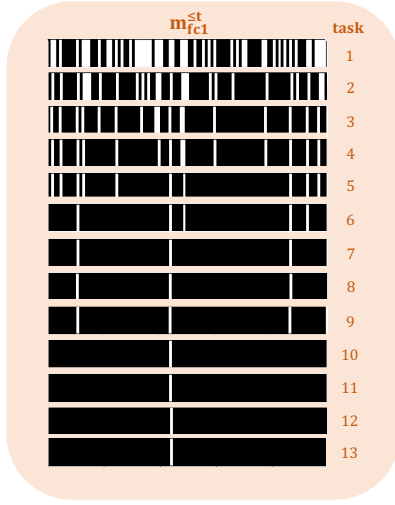
Insufficient network capacity

↓

Reduced learning plasticity
(inbalanced stability-plasticity trade-off)

↓

Drastic performance degradation on new tasks



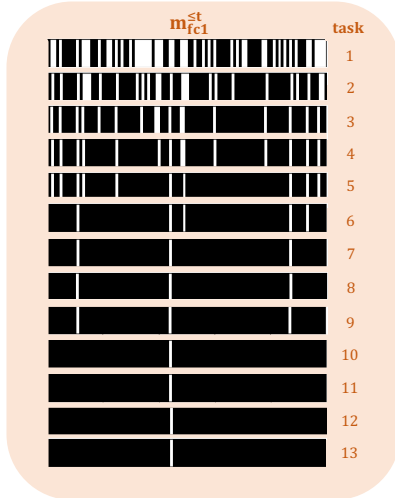
Problem 1: Insufficient Network Capacity

HAT tries to solve it by sparsity regularization on learnable masks:

$$\mathcal{L}' = \mathcal{L}(f(x_t), y_t) + cR\left(\mathsf{M}^t, \mathsf{M}^{<t}\right)$$

$$R\left(\mathsf{M}^t, \mathsf{M}^{<t}\right) = \frac{\sum_{l=1}^{L-1}\sum_{i=1}^{N_l} m_{l,i}^t \left(1 - m_{l,i}^{<t}\right)}{\sum_{l=1}^{L-1}\sum_{i=1}^{N_l}\left(1 - m_{l,i}^{<t}\right)}$$

- Meant to promote higher compactness of the masks and lower network capacity usage
- Helps alleviate the network capacity problem to a certain extent
- However, the network capacity will eventually run out



Problem 2: Non-Adaptive Hyperparameters

Most architecture-based approaches:

- Use several hyperparameters to manually allocate network capacity usage
- Without directly leveraging information about previous tasks
- E.g. PackNet use pruning ratios, HAT uses $s_{\max}$

In continual learning:

- We never know how many tasks in future, maybe infinite ...
- Unable to decide the capacity allocation beforehand
- Manual hyperparameter tuning is difficult

In our work, an adaptive strategy smartly allocates the network capacity with taking into account the information about previous tasks.

AdaHAT: Adaptive Hard Attention to the Task

Our proposed AdaHAT **soft-clips gradients**, which allows 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 directly from HAT architecture:

- **Parameter Importance**
- **Network Sparsity**

AdaHAT: Parameter Importance

The attention vectors (masks) indicate **the importance of parameter**.

Cumulative Attention Vectors (HAT)

$$\mathbf{m}_l^{\leq t} = \max\left(\mathbf{m}_l^t, \mathbf{m}_l^{\leq t-1}\right)$$

- Binary $\{0, 1\}$, represents if it's masked by previous tasks

Summative Attention Vectors (AdaHAT)

$$\mathbf{m}_l^{\leq t,\text{sum}} = \mathbf{m}_l^t + \mathbf{m}_l^{\leq t-1,\text{sum}}$$

- Range from 0 to $t-1$, represents how many previous tasks it's masked
- Contains more information about previous tasks

Adaptive behavior: Higher summative vectors $\rightarrow$ More important to previous tasks $\rightarrow$ Smaller adjustment rate $a^{\star}_{l,ij}$ $\rightarrow$ Smaller updates to the parameter

AdaHAT: Network Sparsity

The sparsity regularization term $R\left(\mathrm{M}^{t}, \mathrm{M}^{<t}\right)$ measures the compactness of masks.

It is closely related to the current network capacity usage:

> *Generally, when a smaller proportion of parameters in the network are static (i.e., sufficient network capacity), the regularization value tends to be larger, as there is a great possibility for the hard attention to be paid to active parameters.*

Adaptive process: Higher sparsity regularization $\rightarrow$ (Suggesting) more unmasked space available for new tasks $\rightarrow$ Less need to adjust the static space for previous tasks, should go for active parameters $\rightarrow$ Smaller adjustment rate $a^{\star}_{l,ij}$ in general

Note: in this way, AdaHAT tries its best to approximate HAT before the network reaches capacity limit, retaining maximal stability before sacrificing plasticity for new tasks.

AdaHAT: The Adjustment Rate

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)

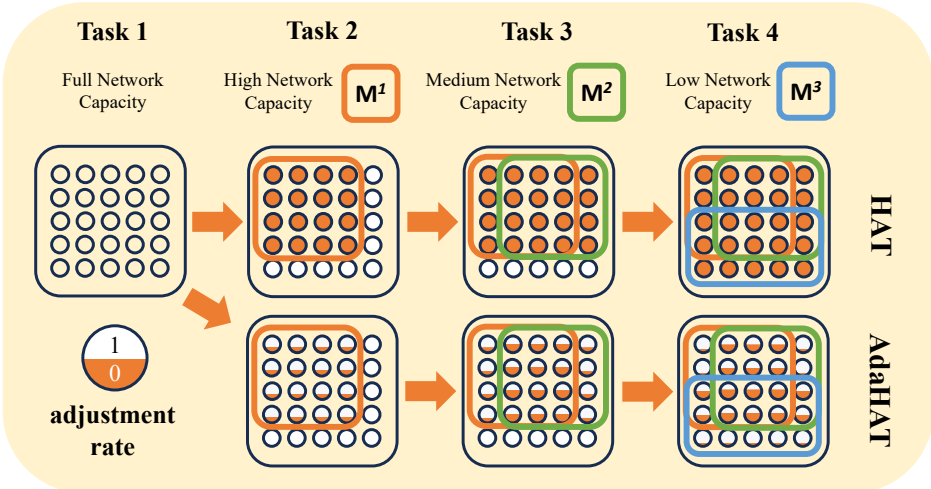
$$a_{l,ij} = 1 - \min\left(m^{<t}_{l,i}, m^{<t}_{l-1,j}\right)$$

The adjustment rate in AdaHAT adaptively incorporates both information about previous tasks:

- Higher parameter importance $\min\left(m^{<t,\mathsf{sum}}_{l,i}, m^{<t,\mathsf{sum}}_{l-1,j}\right)$, lower adjustment rate
- Higher network sparsity $R\left(\mathsf{M}^t, \mathsf{M}^{<t}\right)$, lower adjustment rate

While in HAT only the accumulation of masks.

AdaHAT: Adaptive Hard Attention to the Task



Main Results

Dataset	Approach	AA(↑)	FR (↑)	BWT	FWT
Permuted MNIST	Finetuning	32.62 ± 1.60	-73.78 ± 1.84	-68.10 ± 1.68	0.10 ± 0.04
	Freezing	14.73 ± 0.48	-94.04 ± 0.70	0.00 ± 0.00	-87.13 ± 0.53
	LwF	26.95 ± 1.80	-80.35 ± 2.08	-72.59 ± 1.91	-0.04 ± 0.04
	EWC	52.25 ± 2.46	-51.38 ± 2.83	-42.04 ± 2.67	-8.86 ± 0.09
	HAT	67.64 ± 1.27	-33.70 ± 1.46	-0.11 ± 0.18	-30.54 ± 0.27
	HAT-random	66.43 ± 1.21	-35.10 ± 1.39	-0.27 ± 0.49	-1.47 ± 0.05
	HAT-const-alpha	68.08 ± 1.18	-33.20 ± 1.36	$-1 * e^{-3} \pm 0.00$	-1.39 ± 0.04
	HAT-const-1	48.83 ± 4.35	-55.14 ± 5.02	-49.68 ± 4.40	-3.14 ± 0.07
	AdaHAT	$\mathbf{79.90 \pm 2.40}$	$\mathbf{-19.43 \pm 2.76}$	-14.68 ± 2.48	-2.49 ± 0.06
Split CIFAR-100	Finetuning	24.34 ± 0.73	-91.66 ± 1.32	-54.00 ± 1.00	2.61 ± 0.32
	Freezing	23.22 ± 0.96	-94.73 ± 2.13	0.00 ± 0.00	-57.53 ± 1.20
	LwF	34.56 ± 0.94	-70.91 ± 2.05	-48.03 ± 1.01	4.23 ± 0.42
	EWC	30.23 ± 1.61	-79.84 ± 3.13	-54.05 ± 1.28	-9.80 ± 1.92
	HAT	32.44 ± 1.58	-74.71 ± 3.37	-45.59 ± 1.49	-18.75 ± 0.94
	HAT-random	31.41 ± 1.29	-76.98 ± 2.45	-48.80 ± 1.33	-7.18 ± 1.14
	HAT-const-alpha	32.16 ± 2.48	-75.04 ± 5.16	-44.49 ± 2.57	-5.45 ± 0.82
	HAT-const-1	32.40 ± 1.40	-75.58 ± 3.08	-48.80 ± 1.72	-7.14 ± 1.12
	AdaHAT	$\mathbf{38.74 \pm 2.24}$	$\mathbf{-62.37 \pm 4.64}$	-42.11 ± 2.02	-5.49 ± 0.57

Main Results

Settings:

- Datasets: Permuted MNIST, Split CIFAR-100, 20 tasks
- Metrics for performance:
  - Average Accuracy (AA) over all tasks
  - Forgetting Rate (FR)
- Metrics for stability-plasticity trade-off:
  - Backward Transfer (BWT)
  - Forward Transfer (FWT)

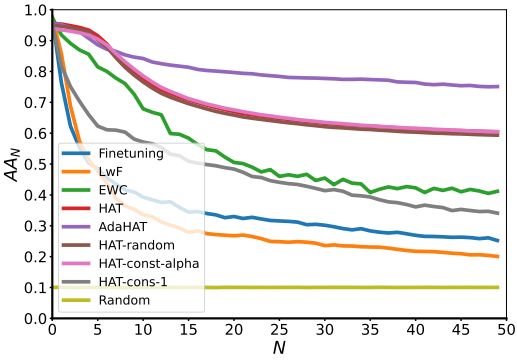
Results:

- AdaHAT outperforms all baselines
- AdaHAT balances stability-plasticity better, while
  - HAT: high BWT, low FWT
  - Finetuning: low BWT, high FWT
  - HAT-const-1: low BWT, high FWT
- Other HAT variants underperform AdaHAT (even HAT)

Conclusions:

- It is important to maintain a balanced stability-plasticity trade-off for optimal performance.
- Gradient adjustment must be properly guided to play a beneficial role.

Results on Longer Task Sequences



Dataset: Permuted MNIST, **50 tasks** (longer)

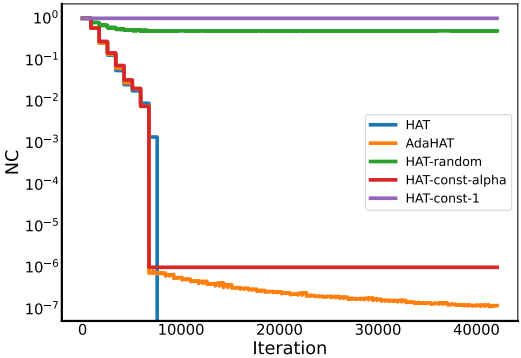
Results:

- HAT slightly outperforms before task 8 then drastically drops
- AdaHAT keeps significant superiority after the turning point
- AdaHAT is still close to HAT before task 8

Conclusions:

- There is a turning point for HAT's performance when it exhausts network capacity
- AdaHAT approximates HAT well before reaching network capacity limit, and shows much more capability for long task sequence settings

Network Capacity Usage



Network Capacity Usage Measure

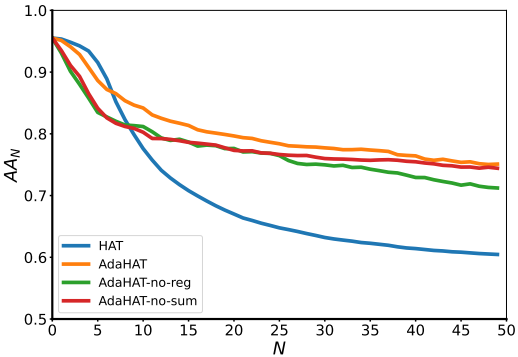
$$\text{NC} = \frac{1}{\sum_{l,i,j} 1} \sum_{l,i,j} a_{l,ij}$$

1 = all parameters can be updated freely
0 = no parameter can be updated

Results and Conclusions:

- HAT runs out of network capacity very soon at a fixed turning point (task 8)
- AdaHAT manages capacity usage **adaptively** over time, making it approach 0 without reaching it

Ablation Study



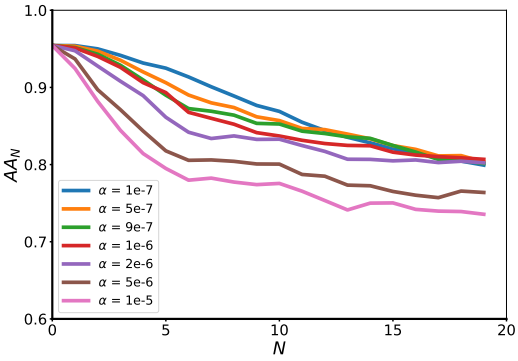
Ablation of two pieces of information:

- **AdaHAT-no-sum:** fix summative $\min\left(m_{l,i}^{<t,\text{sum}}, m_{l-1,j}^{<t,\text{sum}}\right)$ at constant t
- **AdaHAT-no-reg:** fix regularization term $R\left(\mathrm{M}^t, \mathrm{M}^{<t}\right)$ at constant 0

Results: both underperform AdaHAT but outperform HAT

Conclusions: both information play crucial roles for the adaptive extension of HAT.

Hyperparameter Study



AdaHAT introduces only one additional hyperparameter:

- α: overall intensity of gradient adjustment

Results: $\alpha = 10^{-6}$ is optimal

Conclusions:

- Neither small nor large gradient adjustment balances the stability-plasticity trade-off, thus underperforms
- The optimal is still a relatively small value, indicating the importance to design a proper and well-guided adjustment rate

Conclusions

Existing architecture-based approaches (like HAT):

- Tend to tilt the stability-plasticity trade-off towards stability
- Suffer from insufficient network capacity problem in long sequence of tasks

Our proposed **AdaHAT**:

- Balances the trade-off in an adaptive gradient adjustment mechanism, alleviates the network capacity problem
- Also preserve stability benefits before network reaching capacity limit
- Effectively leverages information about previous tasks which was seldom used in architecture-based approaches
- All of them leads to better performance than HAT

Future work: Explore and exploit finer-grained information about previous tasks

Thank You

Thank you for your attention!

Please feel free to ask any questions or reach out to us at:

wangpengxiang@stu.pku.edu.cn

Our work is implemented as a Python package for continual learning research:
pengxiang-wang.com/projects/continual-learning-arena

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