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

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

[1] Peking University, School of Mathematical Sciences, Beijing, China wangpengxiang@stu.pku.edu.cn
[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

Problem Definition

Continual Learning Scenario

Continual Learning (CL): machine learning paradigm, 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 whole 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

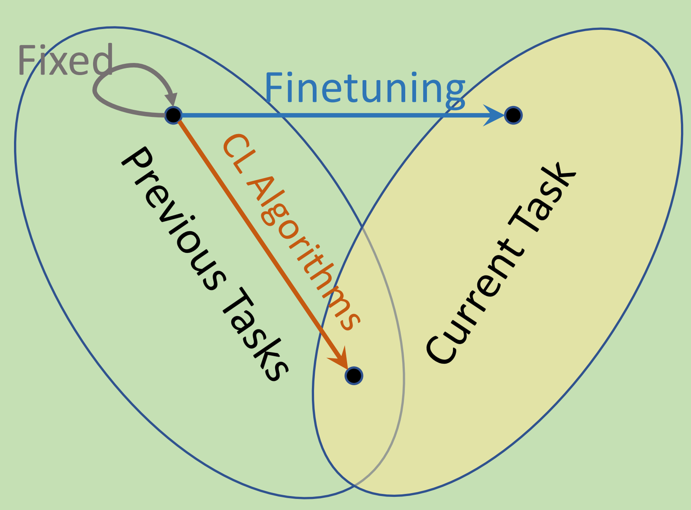
Stability-Plasticity Trade-Off

We must trade off between **stability** and **plasticity**, to get higher performance averaged on all tasks:

	Model Change After New Tasks	Performance
Stability	Not too much	Higher on previous tasks
Plasticity	A lot	Higher on new tasks

Vanilla algorithms (Fixed, Finetuning) break the S-P balance.

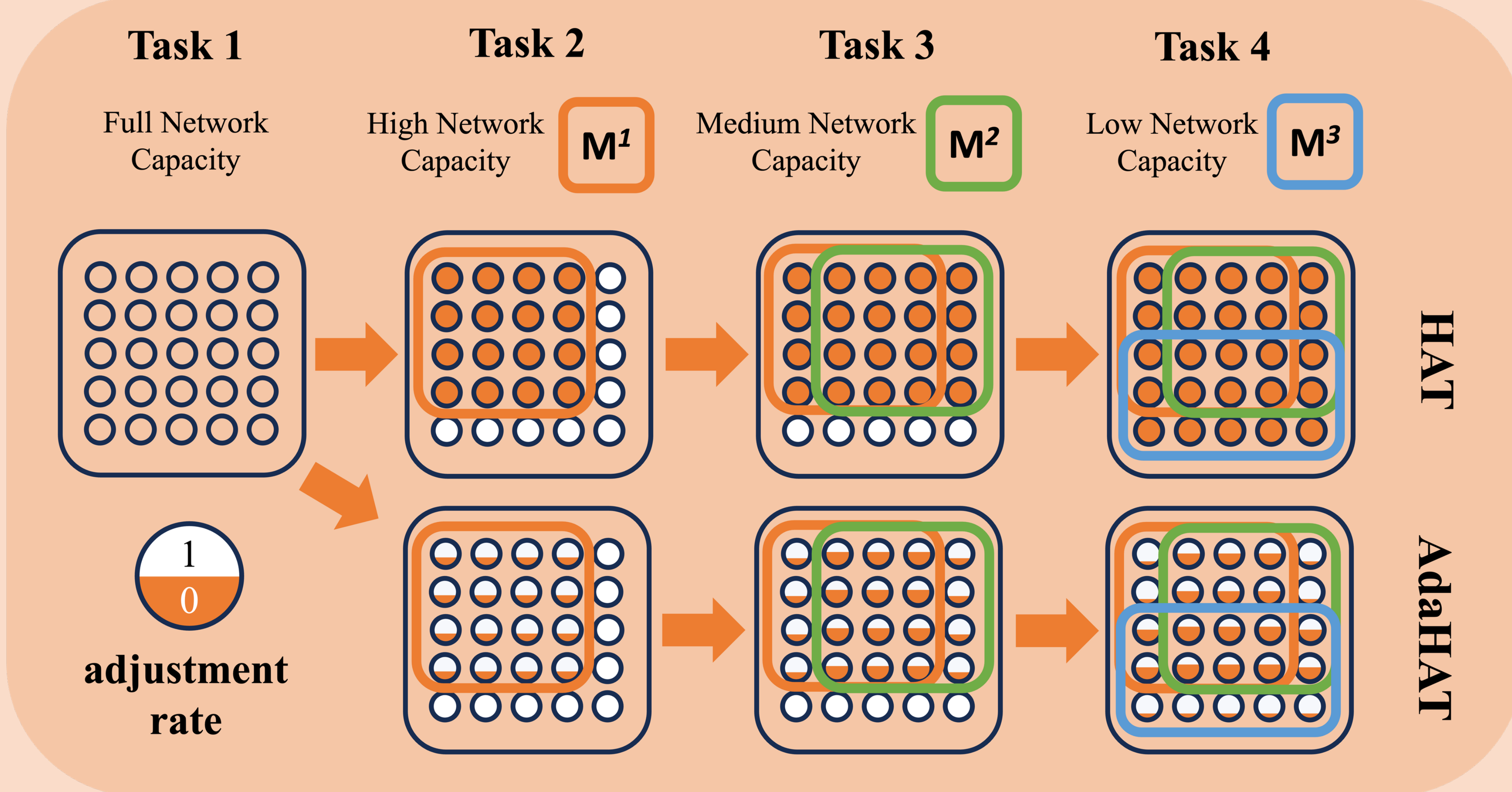
	Mechanism	S-P Balance
Fixed	fix the model after task 1	Too much on stability
Fine-tuning	Init from last task $t-1$	Too much on plasticity **(Catastrophic Forgetting)**



CL Algorithms, such as replay, regularization, gradient-based methods, trade off S-P balance in different ways, using certain form of **information from previous tasks**.

Proposed Approach

We propose **AdaHAT**, an extension to HAT with following mechanisms.



Small Adjustments Allowed

HAT -- hard gradient clipping	**AdaHAT** -- soft gradient clipping
$g'_{l,ij} = a_{l,ij} \cdot g_{l,ij}, a_{l,ij} \in \{0, 1\}$	$g'_{l,ij} = a^*_{l,ij} \cdot g_{l,ij}, a^*_{l,ij} \in [0, 1]$
Either 0 or 1, means whether weights masked by previous tasks	Allow small adjustment on previous tasks with a rate $a^*_{l,ij}$ for more plasticity

$g'_{l,ij}$ is the modified (clipped) gradient from original gradient $g_{l,ij}$ calculated during backpropagation for weight i,j in layer l.

Information Guided Adaptively

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

Adjustment rate uses **information direct from HAT architecture**

- **Parameter Importance:** more previous tasks masked = more important = less adjustment. Indicated by summative mask $m^{<t,sum}_l$
- **Network Sparsity**: more unmasked weights available for new tasks= less need for adjustment. Indicated by mask sparsity reg loss

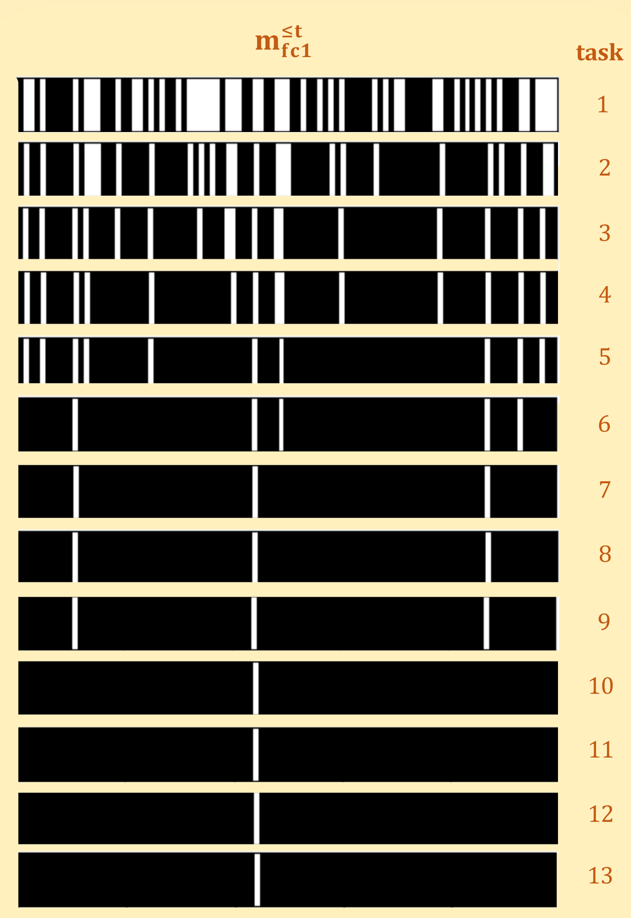
$$R(\mathrm{M}^t, \mathrm{M}^{<t}) = \frac{\sum_l \sum_i m^t_{l,i}(1 - m^{<t}_{l,i})}{\sum_l \sum_i (1 - m^{<t}_{l,i})}$$

Motivation

Limitations on Architecture-based Methods

Architecture-based Methods

- Dedicate parameters in different parts of a network to tasks
- Fix the network parameters learned in previous tasks
- Use network nature, seldom use information from previous tasks
- E.g. **HAT** (Hard Attention to Task)



Limitations on HAT

Parameter space runs out soon after learning a *fixed* number of tasks (usually < 10), that leads to problems:

Insufficient network capacity

- No parameters to learn new tasks
- Sacrifice plasticity for stability
- Perform well on first several tasks, but bad at long sequences of tasks

Not adaptive to task sequences

- Manually tuned hyperparameters to allocate network capacity
- In CL, bear in mind that *we never know how many tasks in future!*

Experiments

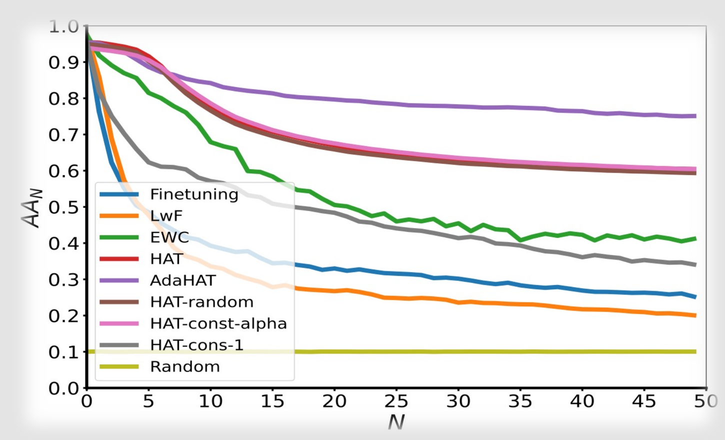
Table 1. performance (mean ± std) and S-P trade-off metrics of different approaches on the two datasets (5 runs, 20 tasks).

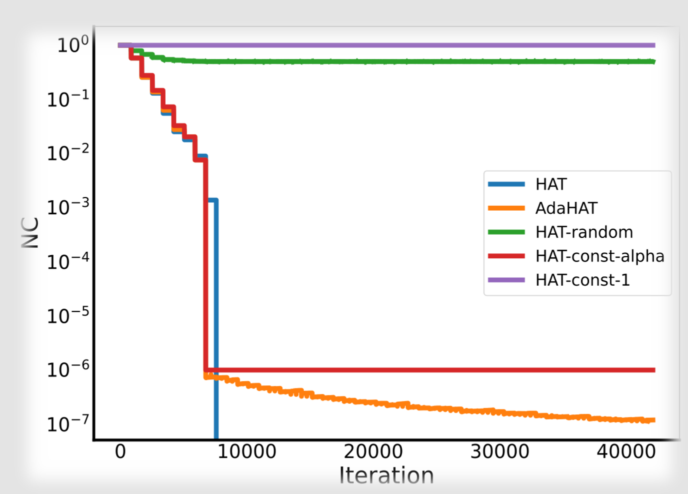
Dataset	Approaches	AA(%)	FR (%)	BWT(%)	FWT (%)
Permuted MNIST	Finetuning (SGD)	32.62 ± 1.60	-73.78 ± 1.84	-68.10 ± 1.68	63.51 ± 0.03
	LwF	26.95 ± 1.80	-80.35 ± 2.08	-72.59 ± 1.91	62.04 ± 0.09
	EWC	52.25 ± 2.46	-51.38 ± 2.83	-42.04 ± 2.67	58.12 ± 0.15
	HAT	67.64 ± 1.27	-33.70 ± 1.46	-0.11 ± 0.18	32.49 ± 1.12
	AdaHAT	$\mathbf{79.90 \pm 2.40}$	$\mathbf{-19.43 \pm 2.76}$	$\mathbf{-14.68 \pm 2.48}$	$\mathbf{59.96 \pm 0.09}$
	HAT-random	66.43 ± 1.21	-35.10 ± 1.39	-0.27 ± 0.49	31.4 ± 1.22
	HAT-const-alpha	68.08 ± 1.18	-33.20 ± 1.36	$-1 * e^{-3} \pm 0.0$	32.92 ± 1.23
	HAT-const-1	48.83 ± 4.35	-55.14 ± 5.02	-49.68 ± 4.4	62.26 ± 0.21
Split CIFAR100	Finetuning (SGD)	24.34 ± 0.73	-91.66 ± 1.32	-54.0 ± 1.0	53.1 ± 0.55
	LwF	34.56 ± 0.94	-70.91 ± 2.05	-48.03 ± 1.01	57.61 ± 0.4
	EWC	30.23 ± 1.61	-79.84 ± 3.13	-54.05 ± 1.28	59.20 ± 0.5
	HAT	32.44 ± 1.58	-74.71 ± 3.37	-45.59 ± 1.49	53.11 ± 0.34
	AdaHAT	$\mathbf{38.74 \pm 2.24}$	$\mathbf{-62.37 \pm 4.64}$	$\mathbf{-42.11 \pm 2.02}$	$\mathbf{56.33 \pm 0.82}$
	HAT-random	31.41 ± 1.29	-76.98 ± 2.45	-48.8 ± 1.33	52.76 ± 0.57
	HAT-const-alpha	32.16 ± 2.48	-75.04 ± 5.16	-44.49 ± 2.57	51.86 ± 0.82
	HAT-const-1	32.4 ± 1.4	-75.58 ± 3.08	-48.8 ± 1.72	56.3 ± 0.36

AA (Average Accuracy over tasks): main performance
Forgetting Ratio (FR): secondary performance
BWT (Backward Transfer): stability metric
FWT (Forward Transfer): plasticity metric

AdaHAT achieves better average performance over tasks ($N = 20$), by a better S-P balance

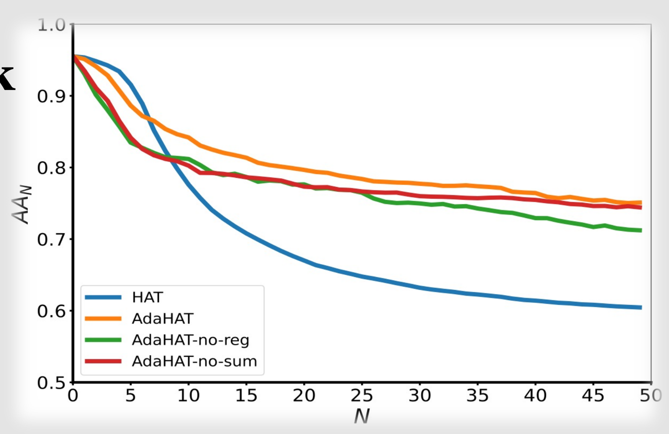
AdaHAT performs well particularly on long sequences of tasks ($N = 50$)





AdaHAT network capacity is consumed in an adaptive way

NC is the average adjustment rate over all static parameters



Ablation study shows both guiding information are vital