CAN MEMORY NETWORKS PLAY A ROLE IN TASK-SPECIFIC MODULATION OF NEURAL CIRCUITS?

Susmit Agrawal
Department of Artificial Intelligence
Indian Institute of Technology Hyderabad
ai22mtech12002@iith.ac.in

Krishn Vishwas Kher
Department of Computer Science
Indian Institute of Technology Hyderabad
cs19b23p000001@iith.ac.in

Madhumitha V
Department of Artificial Intelligence
Indian Institute of Technology Hyderabad
ai23resch11004@iith.ac.in

Vineeth N. Balasubramanian
Department of Computer Science
Indian Institute of Technology Hyderabad
vineethnb@cse.iith.ac.in

ABSTRACT

Neuroscience and artificial intelligence (AI) both grapple with the challenge of adapting neural circuits to diverse tasks while maintaining efficiency and stability. Neuromodulatory systems in the brain dynamically regulate synaptic parameters to enable rapid, context-dependent reconfiguration, mirroring parameter-efficient fine-tuning (**PEFT**) techniques in deep learning. We propose that associative memory (**AM**) mechanisms can serve as a biologically plausible substrate for storing and retrieving task-specific modulatory signals, akin to adapter-based fine-tuning in AI models. Our framework integrates **AM** networks such as Modern Hopfield Networks and Predictive Coding Networks, to store and recall **PEFT**-modulated weights, facilitating task adaptation in task-incremental and multi-task settings. Empirical results on Split-CIFAR100 and Split-TinyImageNet demonstrate that **AM**s can retrieve task-specific modulations with high fidelity, achieving comparable performance to disk-based storage. Our computational experiments show that storing these modulatory signals in **AM**s not only reduces the need for extensive synaptic rewiring but also sheds light on the neural basis of flexible task sets. By bridging neuromodulation and AI memory architectures, our work highlights a shared principle of task-dependent adaptation, offering insights into how the brain may reuse established circuits to meet evolving demands.

1 INTRODUCTION

A central puzzle in neuroscience is how the brain quickly repurposes the same circuitry for multiple tasks without dismantling its core wiring (Duncan, 2001; Miller & Cohen, 2001; Bocincova et al., 2022). Studies of the prefrontal cortex reveal that neural ensembles often display mixed selectivity, where overlapping sets of neurons encode multiple task rules simultaneously (Miller & Cohen, 2001; Rigotti et al., 2013; Warden & Miller, 2010), with neuromodulatory signals regulating the active rule at a given time (Lee & Dan, 2012). Similarly, hippocampal circuits can be rapidly tuned by incoming contextual cues (Preston & Eichenbaum, 2013), thus allowing the same neurons to represent multiple environments or tasks as long as the context is maintained (Muller & Kubie, 1987). From a different perspective, in contemporary artificial intelligence (AI), Large Language Models (Minaee et al., 2024), Vision Transformers (Dosovitskiy et al., 2021), or Large Multimodal Models (Nguyen et al., 2024; Liu et al., 2023) confront an analogous challenge: how to adapt to novel tasks without retraining the entire network. Parameter-Efficient Fine Tuning (**PEFT**) addresses this issue by storing a small number of task-specific parameters instead of modifying the entire parameter set, thus preserving the core model and significantly reducing the computational cost of training.

Neuromodulation in the brain and **PEFT** in deep learning models thus share a common functional principle of selectively overlaying small changes onto a stable parameter backbone. Chemical changes (such as those of dopamine, acetylcholine, and norepinephrine) have been known to transiently adjust synaptic weights or excitability in the cortical and subcortical regions, reshaping their processing modes without destroying prior learning (Lee & Dan, 2012; Doya, 2002). One could view **PEFT** methods such as LoRA (Hu et al., 2021) and its variants (Kopiczko et al., 2024; Gao et al., 2024; Zhang et al., 2023; Hyeon-Woo et al., 2021; Yeh et al., 2024; E.L. Buehler, 2024) as analogous in AI systems, wherein these methods introduce additional adaptable modules that minimally alter forward passes, allowing a shared base model to quickly refocus on a new domain. These parallels underscore the potential for identifying and building on converging principles in biological and artificial systems (Marblestone et al., 2016) on approaches to fast, context-dependent adaptation.

In **PEFT**, particularly in LoRA and its derivatives (Hu et al., 2021; Kopiczko et al., 2024; Gao et al., 2024), low-rank updates are stored in an external memory and retrieved at inference time, enabling a minimal overhead approach to multi-task operations. This observation prompts us to hypothesize that associative memories (AM) can serve as storage mechanisms that support the subsequent on-demand retrieval of such task-specific modulations. A key advantage of such an approach is that associative memory mechanisms may not only represent content (e.g., representation of a specific data instance), but also modulatory information that can be used to dynamically adapt neural circuits to perform specific tasks. In this work, we explore this idea. To this end, we consider recent developments in associative memory networks – Predictive Coding Networks (**PCN**s) (Salvatori et al., 2021; Yoo & Wood, 2022) and Modern Hopfield Networks (Ramsauer et al., 2021) in particular – as a means to store the task-specific modulations for adapting deep learning models. Specifically, we demonstrate this in task-incremental and multi-task learning setups, where adapters are trained for each task and stored in the associative memory. They are then appropriately retrieved at inference for the relevant task. Our experiments show that given relevant cues, such networks can recover the task-specific modulations with high fidelity, achieving performance comparable to accessing such adapter weights from disk.

Our findings in this work have potential implications, both in AI systems and in neuroscience. On the one hand, in the context of AI systems, leveraging associative memory to trigger context-dependent modulatory signals could inspire new architectures that dynamically retrieve task-specific parameter updates for more robust adapter-based learning approaches. Using a memory module can also help in more efficient (or decentralized) use of computational resources for such tasks. From a neuroscience perspective, our framework suggests that neuromodulation in the brain can be approximated as a retrieval process, where task-specific modulations are stored in and recalled from associative memory systems. Such a mechanism also allows postulating a theory for understanding rapid task switching and context-dependent processing observed in regions such as the prefrontal cortex and hippocampus (Hyafil et al., 2009; Preston & Eichenbaum, 2013). Our empirical studies in this work validate that such an approach works in AI systems, where the first task is a meta-task that determines which specific adapter weights need to be modulated over the substrate model for an incoming image, and the second task consists of identifying which class the image belongs to. Experiments on popular datasets such as CIFAR100/TinyImageNet demonstrate the credibility of such an approach.

2 Background and Related Work

Traditional Hopfield networks (Hopfield, 1982) pioneered computational models of associative memories by allowing a set of stored binary patterns to be retrieved via energy minimization, albeit with relatively limited capacity. Recent work on Modern Hopfield Networks (Ramsauer et al., 2021) and universal Hopfield Networks (Millidge et al., 2022) improved upon the original formulation to achieve exponentially greater storage capacity, in addition to enabling storage and retrieval of real-valued vectors. Other approaches use memory-augmented networks, such as Memory Networks (Weston et al., 2015; Sukhbaatar et al., 2015), Neural Turing Machines (Graves et al., 2014), and Differentiable Neural Computers (Graves et al., 2016), integrating explicit read/write operations into an external memory module to support long-range dependency handling and content-based retrieval.

Orthogonally, the surge in large pre-trained models has led to methods that minimize the computational and storage overhead associated with fine-tuning. Techniques like LoRA (Hu et al., 2021) and its variants (Kopiczko et al., 2024; Gao et al., 2024; Zhang et al., 2023; Hyeon-Woo et al., 2021; Yeh et al., 2024; E.L. Buehler, 2024) factorize weight updates into low-rank matrices, while Prefix Tuning (Li & Liang, 2021), Adapter Layers (Houlsby et al., 2019), LayerNorm Tuning (Zhao et al., 2024), and BitFit (Ben Zaken et al., 2022) similarly constrain the number of trainable parameters.

3 METHOD

Our proposed framework uses an external associative memory model to store and retrieve weights. We use task-incremental learning (van de Ven et al., 2022) and multi-task learning (Sun et al., 2021) setups to demonstrate the application and performance of our proposed framework. We use a *memory* as described in the next subsection to store and retrieve adapter weights, and see that bio-inspired memory frameworks can be used to store adapter weights (potentially analogous to storing task-specific modulatory signals in the brain). An overview of the method is shown in Figures 1, 2.

Preliminaries: Let $\mathcal{D}_t = \{(\mathbf{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^N$ represent our dataset for task t, where $\mathbf{x}_i^{(t)} \in \mathcal{X}^{(t)}$ are the input features, $y_i^{(t)} \in \mathcal{Y}^{(t)}$ are the labels, and $t \in \{1, 2, \ldots \mathcal{T}\}$ indicates the task identifier. Each task $\mathcal{T}_t$ is associated with a distinct, non-overlapping class set $\mathcal{Y}_t$. We demonstrate our method in two settings, i.e. task-incremental learning (**TIL**) (van de Ven et al., 2022) and multi-task switching (**MTS**) (Sun et al., 2021). In **TIL**, tasks arrive sequentially, and once a new task begins, all training data samples from previous tasks are lost. However, the task identifier t_i is available during both training and inference. In **MTS**, training data for each task is available at the same time. However, task identifiers are made available only during training and are not provided to the model at inference. Thus in **MTS**, we first infer t_i from the input by a separate predictor whose weights we denote by θ_0. Once t_i is inferred, both settings follow identical pipelines, as they facilitate the substrate model $\mathcal{F}$ to switch contexts (i.e., parameter subsets) accordingly.

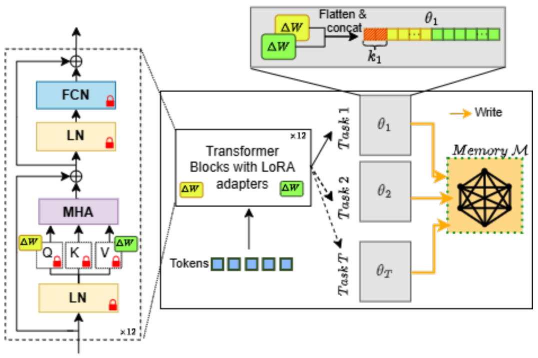

Figure 1: During training, given a dataset for each task t, the base model is trained with task-intrinsic adapters for each such task. In an **MTS** setup, as task predictor is also separately trained to predict task IDs from the aggregate of all the datasets across tasks.

The adapter parameters used for training on task $\mathcal{T}_t$ are denoted as $\theta_{\mathbf{t}}$. The goal is to train/optimize $\mathcal{F}$ to achieve high performance on all tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_t\}$.

3.1 FORMULATION AND IMPLEMENTATION

In our context, a *memory* $\mathcal{M}$ refers to an abstract data type that implements two operations:

Sequential Write, represented by $\mathcal{W}(\cdot)$: This operation stores a provided value in the memory, such that it can be retrieved later using the read operation $\mathcal{R}$. Moreover, the storage should be done in a manner such that the previously written values remain retrievable using an appropriate call to $\mathcal{R}$.

Random Read, represented by $\mathcal{R}(\cdot)$: Given an index (location-based memory) or a query (content-addressable memory), this operation recalls the value stored at the given index or associated with the provided query. Invoking this retrieval operation on a certain value requires it to have already been written to the memory using $\mathcal{W}$.

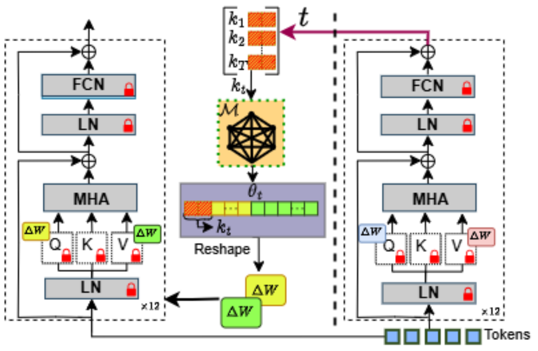

Figure 2: During inference, given the task identifier t, the corresponding modulations are retrieved and overlayed on the base model. In an **MTS** setup, the model is initially configured for the meta-task of predicting the appropriate task identifier for the given input (right of the partition).

We begin by abstracting out the two key steps, storing weights and retrieving them using task identifiers. It should be noted that memory in our case is a reservoir for model weights, not for exemplars (van de Ven et al., 2022) as in prior continual learning approaches that use memories for experience-replay. In an analogous biological model, it would correspond to the engrams that encode a given set of modulatory signals. Of the known biologically plausible **AM** architectures, we identify BayesPCN (Huang & Rao, 2011) and Modern Hopfield Networks (Ramsauer et al., 2021) as candidate architectures that satisfy our definition of a memory. For the remainder of this paper, we represent a generic **AM** model as $\mathcal{M}$, BayesPCN as $\mathcal{M}_{BPCN}$, and the Modern Hopfield Network as $\mathcal{M}_{\mathcal{H}}$.

We train task-specific model-weight modulations θ_t for each task t, using LoRA adapters (Hu et al., 2021) as a proxy for the modulatory signals that are added to the main network. Furthermore, we train the weights of the task ID predictor θ_0 (another set of LoRA adapters) in the **MTS** setting, to allow the model to predict the optimal set of weights to process a given input. After training the model for task t, we compute a task identification vector $\mathbf{k}_t$ and form a tuple $(\mathbf{k}_t, \theta_t)$. In our approach, we set $\mathbf{k}_t$ as a small subset of θ_t itself. Note that this is strictly an engineering choice. We only aim to investigate whether memories can reliably retrieve task-specific modulations given task identification signals; we do not investigate how task identifiers are generated. The memory $\mathcal{M}$ is then updated to store the tuple using a write operation: $\mathcal{M} \leftarrow \mathcal{W}(\mathcal{M}, (\mathbf{k}_t, \theta_t/\mathbf{k}_t))$. During inference, given the task identification vector $\mathbf{k}_t$ (predicted in **MTS** or provided in **TIL**), we can retrieve θ_t from the memory using a read operation: $\hat{\theta}_t \leftarrow \mathcal{R}(\mathcal{M}, \mathbf{k}_t)$. The retrieved modulations may be a close approximation of the original stored ones instead of having the exact values that were previously stored. Hence, we represent them as $\hat{\theta}_t$. The retrieved values $\hat{\theta}_t$ are then loaded into the classifier and the prediction $\hat{y}$ is obtained. The overview of the training & inference pipelines are presented as Algorithms 1 - 4 in Appendix section A.1, respectively.

$\mathcal{W}$ and $\mathcal{R}$ on $\mathcal{M}$: Let $\mathbf{w}_q^{(i)}$ and $\mathbf{w}_v^{(i)}$ be the flattened weights of the adapters modulating the query and value matrices of the i^{th} transformer block in ViT. The weights written to $\mathcal{M}$ at the end of every task are a concatenation of the flattened $\mathbf{w}_q^{(i)}$, $\mathbf{w}_v^{(i)}$ for $i \in [1, 2, 3, ..., L]$, where L is the number of transformer blocks in ViT. Formally, $\theta_t = \mathbf{w}_q^{(1)} \oplus \mathbf{w}_v^{(1)} \oplus \mathbf{w}_q^{(2)} \oplus \mathbf{w}_v^{(2)} \oplus ... \oplus \mathbf{w}_q^{(L)} \oplus \mathbf{w}_v^{(L)}$, where $\oplus$ represents the concatenation operation, for each $t \in \mathcal{T}$. In our implementation, the task identification vector is taken to be the first $|\mathbf{k}_t|$ elements of θ_t, which is a hyperparameter. We set $|\mathbf{k}_t| = \lfloor 0.1 \times |\theta_t| \rfloor$. It should be noted that using part of the modulation signals as queries for associative recall is strictly an engineering choice. In biological systems, $\mathbf{k}_t$ will likely correspond to task-specific neural activity. During inference, there is exactly one forward pass through $\mathcal{F}$ via the weights modulated for the given task ID t in the **TIL** setting, but an additional forward pass in **MTS** setting, since we utilize a different set of modulations over the same substrate network to predict the task ID before retrieving the task-specific modulations.

When using BayesPCN, which is essentially a fully connected network, the $\mathcal{R}$ and $\mathcal{W}$ operations on $\mathcal{M}_{BPCN}$ are formulated using its parameters and its hidden activations, both of which are optimized using Predictive Coding. We kindly refer the reader to the BayesPCN paper (Yoo & Wood, 2022) for further details on how $\mathcal{W}$ and $\mathcal{R}$ are implemented. When using the Modern Hopfield Network instead, a write corresponds to concatenating the task-specific key weights $\mathbf{k}_t$ to a matrix of all keys, $\mathbf{K}$, and concatenating the task-specific value weights $\theta_t/\mathbf{k}_t$ to a matrix of all values, $\mathbf{V}$, as new columns. The read operation then computes the similarity between the provided task-specific query and all stored $\mathbf{w}_q$'s, and uses these similarity scores to perform a weighted combination of all $\mathbf{w}_v$'s. Mathematically, the retrieved parameters $\hat{\theta}_t$ are given by $\hat{\theta}_t = \mathbf{k}_t \oplus \mathbf{V} softmax(\beta \mathbf{K}^T \mathbf{k}_t)$.

4 Experiments and Results

Benchmark Datasets. To evaluate the proposed framework, we perform a comprehensive suite of experiments on Split-CIFAR-100 (Krizhevsky, 2009). We consider 20 and 50 splits (**C100-20S** and **C100-50S**), with 5 and 2 classes per split respectively. We also report results on 50 and 100 splits of the TinyImageNet dataset (Le & Yang, 2015) (**TINet-50S** and **TINet-100S**), containing 4 and 2 classes per split respectively.

Metrics. We use two metrics to evaluate the performance of associative memories in storing and retrieving model weights. The first, Final Average Accuracy (**FAA**), measures the performance of the model by averaging the accuracy across all tasks at the end of the learning process. This metric reflects how well the model retains and applies knowledge acquired from previous tasks when tested

Task	Method	C100-20S	C100-50S	TINet-50S	TINet-100S
	FAA_{GT}	$97.29_{\pm 0.30}$	$97.79_{\pm 0.20}$	$91.56_{\pm 0.48}$	$95.09_{\pm 0.40}$
TIL	$FAA_{TIL}(\mathcal{M}_{BPCN})$	$96.50_{\pm 0.64}$	$97.24_{\pm 0.73}$	$90.46_{\pm 0.35}$	$93.28_{\pm 0.35}$
	$\mathcal{D}_{TIL}(\mathcal{M}_{BPCN})$	$0.32_{\pm 0.12}$	$0.62_{\pm 0.32}$	$1.16_{\pm 0.57}$	$1.71_{\pm 0.20}$
	$FAA_{TIL}(\mathcal{M}_{\mathcal{H}})$	$97.12_{\pm 0.44}$	$97.76_{\pm 0.47}$	$91.82_{\pm 0.28}$	$94.77_{\pm 0.48}$
	$\mathcal{D}_{TIL}(\mathcal{M}_{\mathcal{H}})$	2.6e-3$_{\pm 0.01}$	$0.04_{\pm 0.07}$	$-0.21_{\pm 0.10}$	$-0.30_{\pm 0.19}$
MTS	Task Acc.	$91.47_{\pm 0.21}$	$91.42_{\pm 0.28}$	$84.42_{\pm 0.49}$	$83.27_{\pm 0.18}$
	$FAA_{MTS}(\mathcal{M}_{\mathcal{H}})$	$91.34_{\pm 0.52}$	$94.28_{\pm 0.15}$	$83.14_{\pm 0.16}$	$88.69_{\pm 0.57}$
	$\mathcal{D}_{MTS}(\mathcal{M}_{\mathcal{H}})$	$5.95_{\pm 0.0.22}$	$3.50_{\pm 3.50}$	$8.35_{\pm 0.53}$	$6.47_{\pm 0.73}$

Table 1: Comparison of classification accuracy of recalled weights in **TIL/MTS** against ground truth weights. FAA_{GT} represents the performance using the ground-truth weights. $\mathcal{D}$ indicates the drop in performance when using recalled weights instead of the ground truth. In case of the **MTS** task, the drop includes errors caused due to misidentification of the task an input image belongs to.

on the entire set of tasks. We compute the **FAA** scores over all experiences using the weights recalled from $\mathcal{M}$. Additionally, to compare against the ground truth (**GT**) - loading adapters from disk - we report the difference in **FAA** scores when using weights loaded from disk, represented as FAA_{GT}, versus when recalling the same weights from $\mathcal{M}$, represented as $FAA_{TASK}(\mathcal{M})$. Mathematically, **FAA** is expressed as FAA $= \frac{1}{T}\sum_{t=1}^{T} A_{T,t}$, where T is the total number of tasks, and $A_{T,t}$ is the accuracy on task t after learning the final task T. The difference in FAA scores when loading adapter weights from disk versus loading the weights from $\mathcal{M}$ is denoted by $\mathcal{D}(\mathcal{M}) = FAA_{GT} - FAA_{TASK}(\mathcal{M})$. Together, these metrics provide a comprehensive evaluation of a model's ability to maintain and integrate knowledge over time.

Performance of Associative Memories in storing and recalling adapter weights. In Table 1, we present the results using our benchmark datasets, in **TIL** and **MTS** settings. We find that BayesPCN and Hopfield Networks are both able to recall weights with high fidelity, resulting in very minor drops in performance, if any. In particular, the fidelity of the Hopfield Network is nearly perfect and consistently higher than that of BayesPCN. Interestingly, we find that the performance of weights recalled from the Hopfield Network on TinyImageNet is actually better than the base weights stored on disk - as indicated by the negative value of $\mathcal{D}_{TIL}(\mathcal{M}_{\mathcal{H}})$. We believe that this could be due to cross-task knowledge transfer that happens within the memory during recall, due to the weighted sum of parameters from all tasks. In case of the **MTS** task, we observe that due to the possible error introduced by the task predictor (the accuracy of which is denoted by **Task Acc.**), the ensuing **FAA** score for **MTS** is lower than the corresponding **TIL** score. Nevertheless, our results indicate the feasibility of the framework in implementing dynamic switching of tasks through the use of **AM** storage and retrieval. It should be noted that the drop in performance in **MTS** is not due to **AM** retrieval - it is solely due to errors by the task predictor.

5 Conclusion

This work establishes a conceptual and computational link between neuromodulatory processes in the brain and parameter-efficient adaptation in deep learning. By leveraging associative memory networks to store and retrieve task-specific modulatory signals, we demonstrate a bio-inspired mechanism for efficient task adaptation in task-incremental and multi-task learning. Our experiments show that Modern Hopfield Networks and Predictive Coding Networks can recall adapter weights with minimal degradation, sometimes even improving generalization through implicit knowledge transfer. This insight suggests that neural memories may not only store task-relevant representations but also encode dynamic modulatory patterns essential for rapid adaptation. Future work may extend this framework to class-incremental learning settings, providing another perspective on how the brain can avoid the problem of catastrophic forgetting. It may also lead to refined theoretical models of neuromodulatory recall in biological systems. By uniting perspectives from neuroscience and AI, our framework paves the way for more adaptable and efficient machine learning systems.

References

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL `https://aclanthology.org/2022.acl-short.1/`.

Andrea Bocincova, Timothy J. Buschman, Mark G. Stokes, and Sanjay G. Manohar. Neural signature of flexible coding in prefrontal cortex. *Proceedings of the National Academy of Sciences*, 119 (40):e2200400119, 2022. doi: 10.1073/pnas.2200400119. URL `https://www.pnas.org/doi/abs/10.1073/pnas.2200400119`.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

Kenji Doya. Metalearning and neuromodulation. *Neural Networks*, 15(4):495–506, 2002. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(02)00044-8. URL `https://www.sciencedirect.com/science/article/pii/S0893608002000448`.

John Duncan. An adaptive coding model of neural function in prefrontal cortex. *Nature Reviews Neuroscience*, 2(11):820–829, Nov 2001. ISSN 1471-0048. doi: 10.1038/35097575. URL `https://doi.org/10.1038/35097575`.

M.J. Buehler E.L. Buehler. X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and design. 2024. URL `https://arxiv.org/abs/2402.07148`.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL `http://arxiv.org/abs/1410.5401`.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, Oct 2016. ISSN 1476-4687. doi: 10.1038/nature20101. URL `https://doi.org/10.1038/nature20101`.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/houlsby19a.html`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

Alexandre Hyafil, Christopher Summerfield, and Etienne Koechlin. Two mechanisms for task switching in the prefrontal cortex. *The Journal of Neuroscience*, 29(16):5135–5142, 2009. doi: 10.1523/JNEUROSCI.2828-08.2009.

Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*, 2021.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=NjNfLdxr3A`.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL `https://api.semanticscholar.org/CorpusID:18268744`.

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. URL `https://api.semanticscholar.org/CorpusID:16664790`.

Seung-Hee Lee and Yang Dan. Neuromodulation of brain states. *Neuron*, 76(1):209–222, October 2012.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL `https://aclanthology.org/2021.acl-long.353/`.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

Adam H. Marblestone, Greg Wayne, and Konrad P. Kording. Toward an integration of deep learning and neuroscience, 2016. ISSN 1662-5188. URL `https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2016.00094`.

E K Miller and J D Cohen. An integrative theory of prefrontal cortex function. *Annu Rev Neurosci*, 24:167–202, 2001.

Beren Millidge, Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, and Rafal Bogacz. Universal hopfield networks: A general framework for single-shot associative memory models. In *International Conference on Machine Learning*, pp. 15561–15583. PMLR, 2022.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2024. URL `https://arxiv.org/abs/2402.06196`.

R U Muller and J L Kubie. The effects of changes in the environment on the spatial firing of hippocampal complex-spike cells. *J Neurosci*, 7(7):1951–1968, July 1987.

Thong Nguyen, Yi Bin, Junbin Xiao, Leigang Qu, Yicong Li, Jay Zhangjie Wu, Cong-Duy Nguyen, See-Kiong Ng, and Anh Tuan Luu. Video-language understanding: A survey from model architecture, model training, and data perspectives. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3636–3657, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.217. URL `https://aclanthology.org/2024.findings-acl.217/`.

Alison R. Preston and Howard Eichenbaum. Interplay of hippocampus and prefrontal cortex in memory. *Current Biology*, 23(17):R764–R773, 2013. doi: 10.1016/j.cub.2013.05.041.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=tL89RnzIiCd`.

Mattia Rigotti, Omri Barak, Melissa R. Warden, Xiao-Jing Wang, Nathaniel D. Daw, Earl K. Miller, and Stefano Fusi. The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497 (7451):585–590, May 2013. ISSN 1476-4687. doi: 10.1038/nature12160. URL `https://doi.org/10.1038/nature12160`.

Tommaso Salvatori, Yuhang Song, Yujian Hong, Lei Sha, Simon Frieder, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, 34:3874–3886, 2021.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf`.

Guolei Sun, Thomas Probst, Danda Pani Paudel, Nikola Popović, Menelaos Kanakis, Jagruti Patel, Dengxin Dai, and Luc Van Gool. Task switching network for multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8291–8300, 2021.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, Dec 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00568-3. URL `https://doi.org/10.1038/s42256-022-00568-3`.

Melissa R. Warden and Earl K. Miller. Task-dependent changes in short-term memory in the prefrontal cortex. *Journal of Neuroscience*, 30(47):15801–15810, 2010. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.1569-10.2010. URL `https://www.jneurosci.org/content/30/47/15801`.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. 2015. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083951616&partnerID=40&md5=8194f9dd15bd92b57e7f801973200231`. Cited by: 299.

Shih-Ying Yeh, Yu-Guan Hsieh, Zhidong Gao, Bernard B W Yang, Giyeong Oh, and Yanmin Gong. Navigating text-to-image customization: From lyCORIS fine-tuning to model evaluation. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=wfzXa8e783`.

Jinsoo Yoo and Frank Wood. Bayespcn: A continually learnable predictive coding associative memory. *Advances in Neural Information Processing Systems*, 35:29903–29914, 2022.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=lq62uWRJjiY`.

Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. Tuning layernorm in attention: Towards efficient multi-modal LLM finetuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=YR3ETaElNK`.

A Appendix

Contents

A.1 Training and Inference

In Algorithms 1 and 2, we show the training and inference algorithms in the **TIL** setting used in our proposed framework; analogously, Algorithms 3 and 4 represent the training and inference algorithms in the **MTS** setting. The algorithms explicitly highlight the role of the **AM** in reading and writing weights.

Algorithm 1 TIL Training

```
1: Input: Set of task IDs $\mathcal{T} = \{1, \ldots, T\}$
2: Initialize: Memory $\mathcal{M}_0 \leftarrow \emptyset$,
   Set of task keys $\mathcal{K}_0 \leftarrow \emptyset$
3: for each task ID $t \in \mathcal{T}$ do
4:     Receive dataset $\mathcal{D}_t$ for task $t$
5:     Randomly initialize task-specific parameters $\theta_t$
6:     Update $\theta_t$ based on loss $\mathcal{L}(\theta_t; \mathcal{D}_t)$ using AdamW
7:     Compute task ID vector $\mathbf{v}_t$ from $\theta_t$
8:     $\mathcal{K}_t \leftarrow \mathcal{K}_{t-1} \cup \{\mathbf{v}_t\}$
9:     $\mathcal{M}_t \leftarrow \mathcal{W}(\mathcal{M}_{t-1}, \{\mathbf{v}_t, \theta_t\})$
10: end for
11: Output: Memory $\mathcal{M}_T$, Set of task keys $\mathcal{K}_T$
```

Algorithm 2 TIL Inference

```
1: Input: PEFT Model $\mathcal{C}$, Sample $x$,
   Task ID $\mathbf{v}_t$, Memory $\mathcal{M}_T$
   Intialize: $\mathcal{C} \leftarrow$ base weights $\mathbf{W}$
2: Find Parameters:
3: Retrieve $\hat{\theta}_t \leftarrow \mathcal{R}(\mathcal{M}_T, \mathbf{v}_t)$.
4: Classify:
5: Output $\hat{y} = \mathcal{C}(x, \mathbf{W}, \hat{\theta}_t)$
6: Output: Prediction $\hat{y}$
```

Algorithm 3 MTS Training

```
1: Input: Set of task IDs $\mathcal{T} = \{1, \ldots, T\}$
2: Initialize: Memory $\mathcal{M}_0 \leftarrow \emptyset$,
   Set of task keys $\mathcal{K}_0 \leftarrow \emptyset$
3: Receive datasets $\bigcup_{t \in \mathcal{T}} \mathcal{D}_t$
4: Update $\theta_0$ based on loss $\mathcal{L}(\theta_0; \bigcup_{t \in \mathcal{T}} \mathcal{D}_t)$
5: for each task ID $t \in \mathcal{T}$ do
6:     Randomly initialize task-specific parameters $\theta_t$
7:     Update $\theta_t$ based on loss $\mathcal{L}(\theta_t; \mathcal{D}_t)$ using AdamW
8:     Compute task ID vector $\mathbf{v}_t$ from $\theta_t$
9:     $\mathcal{K}_t \leftarrow \mathcal{K}_{t-1} \cup \{\mathbf{v}_t\}$
10:    $\mathcal{M}_t \leftarrow \mathcal{W}(\mathcal{M}_{t-1}, \{\mathbf{v}_t, \theta_t\})$
11: end for
12: Output: Memory $\mathcal{M}_T$, Set of task IDs $\mathcal{K}_T$
```

Algorithm 4 MTS Inference

```
1: Input: PEFT Model $\mathcal{C}$, Sample $x$,
   Task ID $\mathbf{v}_t$, Memory $\mathcal{M}_T$
   Intialize: $\mathcal{C} \leftarrow$ base weights $\mathbf{W}$
2: Predict task ID:
3: Output $\hat{t} = \mathcal{C}(x, \mathbf{W}, \theta_0)$
4: Find Parameters:
5: Retrieve $\hat{\theta}_{\hat{t}} \leftarrow \mathcal{R}(\mathcal{M}_T, \mathbf{v}_{\hat{t}})$.
6: Classify:
7: Output $\hat{y} = \mathcal{C}(x, \mathbf{W}, \hat{\theta}_{\hat{t}})$
8: Output: Prediction $\hat{y}$
```

A.2 Experimental Details

BayesPCN Hyperparameters. In all your experiments, we use a 4-layer BayesPCN with one particle. The model uses GELU activations in its hidden layers. The learning rate for weight updates during writes is set to 0.0001. For reads, the learning rate is set to 0.01, and the number of read steps is set to 500. Adam optimizer is used for both reads and writes. Any other hyperparameters are set to the default values described in Yoo & Wood (2022).

Hopfield Network Hyperparameters. The only controllable hyperparameter in the MHN module is the value of β, the softmax temperature (Ramsauer et al., 2021). We set this temperature as 25 in all our experiments.

Training Hyperparameters. In all our experiments, we train on each task for 3 epochs with a learning rate starting from 0.005 and decay the learning rate to 0.0001. In the meta-task in the **MTS** setting, we train the task prediction weights for 10 epochs, starting with a learning rate of 0.002 and decaying to 0.0001. We used a weight decay value of 0.1 in all of our experiments, with the AdamW optimizer. In all experiments, we report the mean and standard deviation computed over 3 runs with different seeds. Additionally, we fix the rank of the LoRA adapters to 8 for all experiments in Table 1.

Invariance to LoRA Config: We run our experiments with different configurations of LoRA, changing the low-rank parameter in each configuration. We find that the performance of the recalled weights in each case remains comparable to the performance when using the ground-truth weights. We present the results in Table A.1.

Method/Dataset	LoRA Rank	C100-20S	C100-50S	TInet-50S	TInet-100S
$FAA_{TIL}(\mathcal{M}_{\mathcal{H}})$	4	$96.60_{\pm 0.37}$	$97.70_{\pm 51}$	$91.39_{\pm 0.42}$	$94.94_{\pm 0.59}$
$\mathcal{D}_{TIL}(\mathcal{M}_{\mathcal{H}})$	4	$0.0_{\pm 0.05}$	$-0.03_{\pm 0.0}$	$-0.1_{\pm 0.24}$	$0.09_{\pm 0.31}$
$FAA_{TIL}(\mathcal{M}_{\mathcal{H}})$	8	$97.12_{\pm 0.44}$	$97.76_{\pm 0.47}$	$91.82_{\pm 0.28}$	$94.77_{\pm 0.48}$
$\mathcal{D}_{TIL}(\mathcal{M}_{\mathcal{H}})$	8	$2.6\text{e-}3_{\pm 0.01}$	$0.04_{\pm 0.07}$	$-0.21_{\pm 0.10}$	$-0.30_{\pm 0.19}$
$FAA_{TIL}(\mathcal{M}_{\mathcal{H}})$	32	$96.99_{\pm 0.01}$	$97.62_{\pm 0.44}$	$91.64_{\pm 0.44}$	$95.09_{\pm 0.47}$
$\mathcal{D}_{TIL}(\mathcal{M}_{\mathcal{H}})$	32	$0_{\pm 0.01}$	$0.20_{\pm 0.02}$	$-0.04_{\pm 0.34}$	$0.13_{\pm 0.47}$

Table A1: Comparison of classification accuracy of recalled weights in **TIL** against ground truth weights, when the low-rank parameter of the adapters is varied.

Training Hardware. All experiments were executed on a single A6000 Ada 48GB GPU coupled with a 96-core CPU. Each model that uses Hopfield Networks takes between $2 - 4$ hours to train on all experiences, depending on the number of splits and training iterations. Experiments with BayesPCN take up to a day for TinyImageNet-100 Split.

A.3 Limitations

This work intends to highlight the strong parallels that exist between the two fields of rapid task switching in Neuroscience, and Parameter-Efficient Fine Tuning in Deep Learning, that have been traditionally looked at independently. Additionally, it proposes a potential mechanism in which such task-switching can occur in biological systems with the aid of associative memories and provides a proof-of-concept. As such, it does not contain experiments on a large number of datasets or variants of LoRA. More importantly, all the experiments provided are based on computational models from deep learning research. For sound evidence, analogous experiments from the neuroscience community will be required to validate the hypothesis proposed.