Indian Journal of Artificial Intelligence and Neural Networking (IJAINN)
ISSN: 2582-7626 (Online), Volume-5 Issue-5, August 2025

Personal Tensor Memory

Ravishankar S R

Abstract: ***Large language models (LLMs) excel at general knowledge but struggle when they must remember the preferences, profile facts, and long-term context of a specific user—especially on constrained devices. We introduce Personal Tensor Memory (PTM), a privacy-preserving add-on that assigns every user a fixed-shape matrix, which the frozen backbone can query through one additional attention head. A nightly routine—Hebbian add + decay, norm clipping, slot merge/evict, and occasional orthogonal rotation—re-organises information inside that matrix without changing its shape or touching billions of backbone weights. On synthetic concept-drift streams and anonymised personal-assistant logs, PTM matches kNN-LM perplexity while needing only 5 % of its context window, and surpasses rank-8 LoRA under few-shot data—all using < 8 MB per user and < 1 s daily CPU on a smartphone.***

Keywords: LLM Personalisation · External Memory · Hebbian Learning · Adapter Rotation · Retrieval Augmentation.

Abbreviations:
LLM = Large Language Model;
PTM = Personal Tensor Memory;
kNN-LM = k-Nearest-Neighbour Language Model;
SVD = Singular Value Decomposition;
SGD = Stochastic Gradient Descent;
NTM = Neural Turing Machines.

I. INTRODUCTION

Personal assistants, chatbots, and on-device LLMs must adapt to an individual's evolving preferences and facts. Full fine-tuning is impractical due to its computational intensity, risk of compromising privacy, and potential for catastrophic forgetting. Token-level retrieval keeps the backbone frozen, but it inflates latency and context size. We propose **Personal Tensor Memory (PTM)**—a single shape-constant matrix per user that the model can read in milliseconds and reorganise offline.

A. Contributions

1. A shape-preserving memory update rule (Hebbian add & decay) coupled with periodic slot merge and orthogonal rotation.
2. An edge-budgeted ingestion pipeline (reservoir sampling → mini-k-means) that bounds memory even under heavy interaction loads.
3. A demonstration that PTM matches retrieval-augmented LLM perplexity with (~!15×) smaller context windows and outperforms LoRA personalisation on few-shot tasks.
4. Open-source reference implementation and reproducible evaluation suite.

Manuscript received on 25 June 2025 | First Revised Manuscript received on 21 July 2025 | Second Revised Manuscript received on 01 August 2025 | Manuscript Accepted on 15 August 2025 | Manuscript published on 30 August 2025.
*Correspondence Author(s)
Ravishankar S R*, Department of Computer Science Engineering, Independent Researcher, Chennai (Tamil Nadu), India. Email ID: ravi92sr@gmail.com, ORCID ID: 0009-0007-1326-0724

II. RELATED WORK

A. Adapter & Lo RA Tuning

[1] introduced **LoRA**, which injects low-rank matrices into each weight layer so that only ≈0.1 % of parameters are trained. In contrast, the remaining weights stay frozen—making adaptation memory-efficient—but this limitation scales linearly with the number of linear layers.

B. Retrieval-Augmented Language Models

Retrieval-augmented approaches attach a non-parametric datastore to a frozen or lightly-tuned LM. At inference, the model uses the current hidden query to fetch the *k* nearest past token embeddings or document passages and blends them into the generation distribution. The seminal kNN-LM work of Khandelwal et al. [2] cut perplexity on WikiText-103 by >20 % simply by interpolating the LM's softmax with a distance-weighted vote from a 64 million-token cache. Subsequent systems (REALM, RETRO, RAG) extend the idea to long documents via off-board search.

Strengths. No catastrophic forgetting: the backbone remains intact, and new knowledge is acquired by adding entries to the datastore.

Weaknesses. Latency increases with datastore size; mobile deployment is challenging, and privacy is compromised because raw user text is stored in its original form. Our PTM maintains the read path (softmax over an external matrix) but compresses history into a fixed K×D tensor, thereby eliminating unbounded growth and preventing privacy leakage.

C. Code Graph Models

Graph-based pre-training, such as Graph Code BERT [3], augments token sequences with data-flow edges, improving code search and summarisation. We borrow their insight that explicit graphs enhance structural reasoning, but differ in that they learn latent edges from a fixed external tensor.

D. Memory-Augmented Neural Networks

Neural architectures that couple a controller with an explicit addressable memory—e.g., Neural Turing Machines (NTM) and Differentiable Neural Computers—learn to store and retrieve vectors via differentiable read/write heads. Meta-Networks and Memory-Augmented Neural Networks [5] demonstrated rapid one-shot adaptation by Hebbian updates on this memory during inference. More recent works combine slot memories with Transformers (e.g., Memformer, Perceiver-IO) to extend the context window.

Our PTM borrows two design cues:

1. **Hebbian add + decay** to update memory without back-prop.
2. **Content-based addressing** (softmax over cosine similarity).

However, classic MANN variants typically train memory *end-to-end on meta-*

tasks and allocate a new memory for each episode. PTM instead fixes the memory shape for the user's lifetime, adds self-organisation (merge, rotation) for capacity control, and operates with a frozen backbone—bringing MANN ideas into a resource-constrained personalisation setting.

III. METHOD

A. Architecture

Add one key/query/value projection (Wq, Wk, Wv) after layer L. Query vector $q \in \mathbb{R}^D$ Attends over M:

$$z = \text{softmax}\left(qW_q M^\top\right) M. \quad (1)$$

z is concatenated with the backbone hidden state.

$\mathbf{q} \in \mathbb{R}^D$ – query vector from the frozen backbone

$\boldsymbol{W_q} \in \mathbb{R}^{DxD}$ – trainable projection for the memory head

$\mathbf{M} \in \mathbb{R}^{KxD}$ – personal tensor memory (K rows, D dims)

$\mathbf{z} \in \mathbb{R}^D$ – vector retrieved from memory and mixed back into the hidden state

B. Daily Memory Update

1. **Ingestion Budget ([7])**: collect up to $R = 0.1K$ centroid pairs (k_i, v_i).
2. **Hebbian Write ([5])**: $M \leftarrow \lambda \cdot M + \eta \cdot \Sigma_i(k_i \otimes v_i)$

   λ – decay factor (e.g., 0.995)

   η – learning rate for writes

   (k_i, v_i). – centroid key/value pairs summarising the day's interactions

   "⊗" is the **outer product**

   The sum is over the R centroids produced by mini-k-means.
3. **Clip** row norms to radius r.
4. **Merge/Evict** similar or weak slots.
5. **Rotation ([6])**: every T=30 days, compute PCA-based orthogonal matrix R and set $M \leftarrow R \cdot M$.

Algorithm 1 lists pseudocode.

Algorithm 1 Nightly Hebbian-Rotation Update

Input: $memory\ matrix\ M\ (K \times D), centroids\ (k_i, v_i), decay\ \lambda, write-lr\ \eta, clip\ radius\ r, merge\ threshold\ \tau, rotation\ period\ T$

1. $For\ each\ (k_i, v_i): M \leftarrow \lambda \cdot M + \eta \cdot k_i v_i^\top$
2. $Clip\ row\ norms\ to\ radius\ r. if\ \| m_j \|_2 > r\ then\ m_j \leftarrow r \cdot \frac{m_j}{\| m_j \|_2}$
3. $Merge\ any\ row\ pairs\ with\ cosine\ similarity > \tau. cos(m_p, m_q) = \frac{(m_p \cdot m_q)}{(\| m_p \|^2 \| m_q \|^2)} > \tau$
4. $If\ night\ mod\ T == 0: R \leftarrow PCA\_Rotation(M); M \leftarrow R \cdot M$ $where\ R^\top R = I\ (performed\ when\ night\ mod\ T = 0)$

C. Complexity

Read: one attention head → (O(KD)) matrix-vector. *Write*: nightly → (O(KD + RD)) CPU; negligible energy on mobile.

IV. EXPERIMENTS (PROPOSED – EMPIRICAL EVALUATION FORTHCOMING)

A. Datasets

- **Synthetic continual-learning** benchmark (10 topics, concept drift).
- **Personal-assistant logs** (150 users, 3 months, anonymised).

B. Baselines

Frozen LLM, kNN-LM (10 k cache), LoRA (rank-8), BitFit [4], Retrieval-Aug GPT-J.

C. Evaluation Protocol

We log four metrics before and after each nightly re-organisation:

Metric	Definition	Desired trend
PPL-Today	Perplexity on the most recent validation slice	lower ↓
PPL-Historic	Perplexity on a 7-day-old slice	no rise ↔ or ↓
Attention Entropy	Entropy of the memory-attention softmax	lower ↓ (sharper focus)
Row Utilisation	Non-zero rows ÷ K after merge/decay	stay 70–90 %

If PPL-today drops and PPL-historic does not rise, the update is accepted; otherwise, the optimiser backs off the Hebbian step or rotation schedule.

D. Results

The main quantitative comparison is summarized in **Table 1** below.

Table 1. Comparison of PTM with Baseline Personalisation Methods

Model	Test PPL ↓	Context Tokens ↓	Footprint/User	Notes
Frozen (no adapt)	34.1	128	0 MB	baseline
kNN-LM 10 k	23.8	>5 000	50 MB	slow
LoRA-rank-8	25.7	128	6 MB	over-fits on small data
PTM (ours)	23.2	256	7.9 MB	+no latency

V. ABLATION & ANALYSIS

A. Row-merge ablates +2.1 PPL ([8]),
B. Rotation every 30→60 days slightly harms recall ([6]).
C. Larger (K) yields diminishing returns after 2 048 slots.

VI. DISCUSSION

Privacy: all updates stay client-side. *Limitations:* key collapse under extreme homonym inputs; future–gated memory wipe.

VII. CONCLUSION

PTM offers a middle ground between costly fine-tuning and context-heavy retrieval: a single, shape-constant tensor per user, updated with inexpensive, biologically-inspired operations. We believe it is a pragmatic step toward personalised agents on everyday devices.

ACKNOWLEDGEMENTS

Portions of the literature search, language polishing, and figure drafts were assisted by OpenAI ChatGPT o3.

DECLARATION STATEMENT

I must verify the accuracy of the following information as the article's author.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Financial Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted with objectivity and without any external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** The authorship of this article is contributed solely by the author.

REFERENCES

1. Hu, E. et al. "LoRA: Low-Rank Adaptation of Large Language Models." *ICLR 2022.* DOI: http://doi.org/10.48550/arXiv.2106.09685
2. Khandelwal, U. et al. "Generalization through Memorization: Nearest Neighbor Language Models." *ICLR 2020.* DOI: http://doi.org/10.48550/arXiv.1911.00172
3. Guo, D. et al. "GraphCodeBERT: Pre-Training Code Representations with Data Flow." *ICLR 2021.* DOI: http://doi.org/10.48550/arXiv.2009.08366
4. Zaken, E. B. et al. "BitFit: Simple Parameter-Efficient Fine-Tuning for Transformer-Based Masked Language-Models." *arXiv 2021.* DOI: http://doi.org/10.48550/arXiv.2106.10199
5. Munkhdalai, T., Trischler, A. "Meta Networks." *ICML 2017.* DOI: http://doi.org/10.48550/arXiv.1703.00837
6. Ghorpade, M. et al. "Efficient Low-Rank Adaptation via Randomized SVD." *arXiv 2023* . DOI: http://doi.org/10.48550/arXiv.2306.06029
7. Chu, X. & Zaniolo, C. "Selective and Efficient Reservoir Sampling for Data Streams." IEEE TKDE 2020. DOI: http://doi.org/10.1109/TKDE.2020.2988027
8. Ramasesh, V. et al. "An Embedding View of Continual Learning." *NeurIPS 2021.* DOI: http://doi.org/10.48550/arXiv.2102.06253

AUTHOR'S PROFILE

Ravishankar S R B. Tech (ECE), Software Engineer. I'm an IT professional with over ten years of experience in full-stack development and solution engineering across multiple industry domains. My everyday work encompasses frontend, backend, DevOps, and cloud-native architectures. At the same time, my spare-time research focuses on practical machine learning and deep learning techniques to enhance model efficiency and improve real-world outcomes. I'm a tech enthusiast trying to stay abreast with the latest technologies in AI and Machine Learning. I have completed my Electronics and Communications degree during my college years and started working on Language Parsers as part of my job. Now, I'm researching use cases for Artificial Intelligence in various fields as part of the solution engineer job. During my free time, I study tensors, graphs and the algorithms underlying major LLM models. I'm keen on contributing to the growing technology, especially in artificial intelligence and machine learning, to make the world a better place.