Comparative Analysis of Internal Representations in Hebbian Learning Algorithms

Dumitru-Sebastian Mustaţă

Supervisor(s): Stephanie Tan, Yaqi Guo | Committee: Inald Lagendijk

EEMCS, Delft University of Technology, The Netherlands

1 Motivation

- BP suffers from the **weight transport problem**, **update locking**, and high **memory overhead** [1] – motivating the search for alternatives.
- **Hebbian learning** ("neurons that fire together, wire together") offers a biologically plausible, local alternative.
- Representational analyses are widely used to compare *architectures*, but *BP-free learning algorithms* remain a "black box".

► Same architecture, different learning rules – what changes in the internal representations?

2 Research questions

How do internal representations, feature extraction, and computational efficiencies of forward-only, localised learning compare to standard Backpropagation in visual classification tasks?

SQ1 – Geometry & Similarity: How similar are their hidden spaces, and how separable is the data?

SQ2 – Feature Extraction: What visual patterns drive neuron activations?

SQ3 – Hardware Evaluation: Does theoretical efficiency translate to real benefits?

3 Three learning paradigms

Backpropagation (Baseline): Global error propagated *backwards* via the chain rule. Not local, not parallelisable.

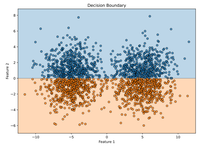
PEPITA [2]: Forward-only, supervised, derives local updates. Replaces backward pass with an *error-modulated* second forward.

SoftHebb [3]: Unsupervised Hebbian plasticity with soft winner-take-all. No labels, no error signal.

► How does PEPITA relate to Hebbian Learning? Can SoftHebb overcome its unsupervised nature? And how do they compare to BP?

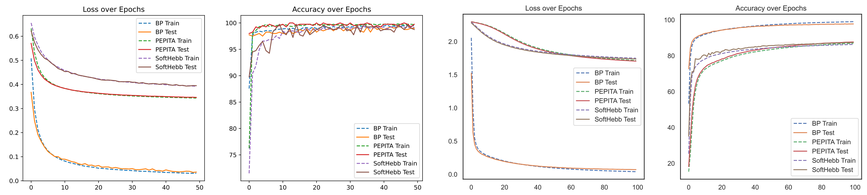
4 Datasets & setup

Dataset	Dim	Hidden	Classes	Samples	Epochs
Synthetic	2	64	2	2 000	50
MNIST	784	512	10	70 000	100



- 3-layer MLPs; SGD optimiser.
- Analyses: linear probing [4], PCA [5], CKA [6], saliency mapping [7].

5 Synthetic dataset & MNIST accuracy

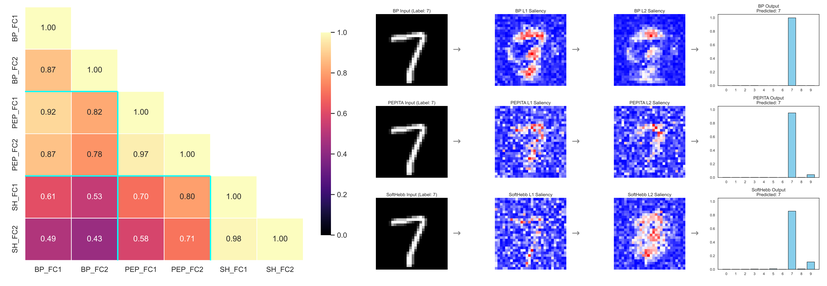


- **Synthetic:** All three algorithms override visual clusters to solve the classification with ∼99% accuracy – even SoftHebb, despite being unsupervised.
- **MNIST:** BP achieves 97.75%; PEPITA and SoftHebb ∼87%.

► However, linear probing on frozen hidden layers yields ∼94% for both PEPITA and SoftHebb, exposing a bottleneck in the classifier, not the representations.

6 Representational dynamics: PCA, CKA & saliency (SQ1 & SQ2)

- **PCA:** BP disentangles classes cleanly; SoftHebb encodes *structural* variance rather than class identity, yet remains linearly separable in higher dimensions.



- **CKA:** PEPITA's shallow layers mirror BP (0.92), but its deeper layers converge toward SoftHebb (0.80) – the error perturbation diffuses in deeper layers.
- **Saliency:** BP reveals sharp, concentrated activations on discriminative edges; SoftHebb closely follows the digit's shape, but with lower certainty on the output class.

7 The hardware efficiency (SQ3)

Metric	BP	PEPITA	SoftHebb
Avg. sec / epoch	**3.64**	5.00	4.62
Peak memory (MB)	**420**	466	450

- **Counter-intuitive:** BP-free methods are theoretically lighter, yet run *slower* and use *more memory* in practice.
- **Root cause:** Frameworks like PyTorch are heavily optimised for BP; custom algorithms implemented directly in Python experience substantial software overhead.

► The bottleneck lies in standard software frameworks, not the theoretical efficiency of the algorithms.

8 Key take-aways

1. **BP** sets the upper bound for class separation.
2. **SoftHebb** representations are richer than end-to-end accuracy suggests.
3. **PEPITA** is a bridge: gradient-like topology in shallow layers, balancing with unsupervised representations in deeper layers.
4. The **accuracy gap** (∼87% vs. ∼94% with linear probing) is a classifier bottleneck, not a representational one.

9 Limitations & future work

- **Dataset complexity:** Validate on CIFAR-10 to confirm PEPITA's bridging behaviour generalises.
- **Native implementations:** Re-implement BP-free rules as low-level C++ extensions to benchmark true efficiency.
- **Hybrid architectures:** SoftHebb for unsupervised feature extraction and PEPITA for task-specific classification.
- **Neuromorphic deployment:** Deploy on physical neuromorphic hardware to measure efficiency.

References

[1] R. Ye et al. *TechRxiv* (2026).
[2] G Dellaferrera et al. *arXiv preprint arXiv:2201.11665* (2022).
[3] T. Moraitis et al. *Neuromorphic computing and engineering* (2022).
[4] G. Alain et al. *arXiv preprint arXiv:1610.01644* (2016).
[5] K. Pearson. *The London, Edinburgh, and Dublin philosophical magazine and journal of science* (1901).
[6] S. Kornblith et al. *International conference on machine learning*. PMIR. 2019.
[7] K. Simonyan et al. *arXiv preprint arXiv:1312.6034* (2013).