



Comparative Analysis of Internal Representations in Hebbian Learning Algorithms

Dumitru-Sebastian Mustață

Supervisor(s): Stephanie Tan, Yaqi Guo

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

June 26, 2026

Name of the student: Dumitru-Sebastian Mustață
Final project course: CSE3000 Research Project
Thesis committee: Stephanie Tan, Yaqi Guo, Inald Lagendijk

Comparative Analysis of Internal Representations in Hebbian Learning Algorithms

Dumitru-Sebastian Mustăță
Delft University of Technology

Stephanie Tan, Yaqi Guo
Delft University of Technology

Abstract

The backpropagation (BP) algorithm, though fundamental to modern deep learning, faces severe biological, computational, and physical limitations that hinder its applicability on energy-efficient neuromorphic systems. This has motivated the search for BP-free learning paradigms, with Hebbian-based algorithms being a notable alternative. Existing approaches range from purely local, unsupervised schemes such as SoftHebb—which naturally clusters data based on structural variance—to supervised, error-modulated approaches like PEPITA, which provides task-specific feedback through a second forward pass. However, the trade-offs between these extremes remain largely unexplored. This study systematically compares the internal representations and hardware efficiencies of three Multi-Layer Perceptrons (MLP) trained with Backpropagation, PEPITA, and SoftHebb.

Our evaluation, utilizing geometric metrics such as Centered Kernel Alignment (CKA) and Principal Component Analysis (PCA), showcases how PEPITA exhibits higher similarity to BP, but is more geometrically aligned with the unsupervised SoftHebb. Furthermore, empirical hardware profiling exposes a significant implementation paradox: despite the theoretical efficiency of BP-free methods, high-level framework bottlenecks currently make algorithms like PEPITA computationally expensive on traditional digital architectures.

1 Introduction

Backpropagation (BP) is the foundational training algorithm for modern deep learning, but its continued scaling has exposed significant computational and biological limitations. From a biological perspective, BP suffers from the *weight transport problem* [1], as biological synapses lack the mechanism to instantly transpose weight matrices. Computationally, BP encounters the *update locking problem* [2], requiring each forward and backward pass to finish before the next can begin. Additionally, BP poses a significant *memory bottleneck* because it must store all intermediate activations during the forward pass to compute gradients later [3].

These limitations, combined with the physical constraints of traditional chip manufacturing, motivate the search for alternative, forward-only learning paradigms. This is particularly relevant for deployment on emerging neuromorphic hardware and physical neural networks. While several BP-free algorithms have been proposed, the current literature primarily evaluates them based on classification accuracy. Representational similarity metrics are typically reserved for comparing distinct deep learning architectures (e.g., CNNs vs. Vision Transformers [4]). However, understanding how BP-free algorithms internally disentangle data is

crucial for assessing their robustness to adversarial noise, evaluating their out-of-distribution generalization, and future designing of hybrid models.

To explore this, we focus on Hebbian learning (HL), which is based on the biological principle that "neurons that fire together, wire together" [5]. We select two specific BP-free algorithms: SoftHebb [6] and PEPITA [7]. SoftHebb is chosen for its strict adherence to local, single-phase, unsupervised Hebbian plasticity, providing a baseline for how networks cluster data based solely on structural variance. For the supervised approach, we selected PEPITA. It eliminates the backward pass by using an error-modulated second forward pass to guide local updates. What makes PEPITA particularly intriguing is its relationship to Hebbian learning. While it deviates from the classic unsupervised "neurons that fire together, wire together" paradigm, it remains mathematically rooted in Hebbian principles by computing weight updates derived strictly from local pre- and post-synaptic activations.

The central aim of this thesis is to open the "black box" of these localized learning rules. While the scientific community has spent the last decade developing sophisticated tools to analyse standard, gradient-based neural networks, the properties of these new, localized paradigms remain largely unexamined. We understand the mathematical rules governing SoftHebb and PEPITA, but how these local updates organize into global geometric structures and semantic feature representations is currently unknown. Therefore, we systematically compare purely local (SoftHebb), forward-only supervised (PEPITA), and standard gradient-based (BP) learning to address our primary research question: *How do the internal representations, feature extraction strategies, and computational efficiencies of forward-only and localized learning rules compare to standard Backpropagation in visual classification tasks?*

To address this, our investigation is divided into three specific sub-questions:

SQ1: Geometric and Similarity Analysis How similar are their internal representations to one another, and how do these learning rules differ in their ability to separate and group data?

SQ2: Feature Extraction What specific visual patterns and features in the input images trigger neuron activations in each of these algorithms?

SQ3: Hardware Evaluation Do the theoretical efficiency benefits of BP-free methods actually result in lower memory usage and faster execution times on standard hardware?

To investigate representational structures (SQ1), we employ Principal Component Analysis (PCA) [8] and Centered Kernel Alignment (CKA) [9] to demonstrate how supervised forward passes act as a bridge between gradient-based disentanglement and unsupervised clustering. For feature extraction (SQ2), we apply gradient-based saliency mapping [10] to intermediate activations, shifting the focus from *what* the network predicts to *why* it predicts it by highlighting the specific input elements driving neuronal responses. Finally, in our hardware evaluation (SQ3), we benchmark execution time and memory usage, exposing a critical implementation paradox: despite their theoretical efficiency, high-level software frameworks impose constraints that currently bottleneck localized algorithms on standard digital architectures.

Ultimately, our findings show that while BP remains the upper bound for mathematically optimal class separation, PEPITA successfully balances a gradient-like topology with SoftHebb-like structural variance. Furthermore, despite current software-level bottlenecks, the competitive performance of these BP-free models positions them as highly viable candidates for parallelization and future neuromorphic deployment.

By addressing these three inquiries, this study provides a systematic representational comparison of forward-only learning paradigms, serving as a foundational step towards the realization of scalable, energy-efficient neuromorphic AI.

The remainder of this paper is structured as follows: Section 2 provides the background on the evaluated algorithms. Section 3 outlines the analytical framework and evaluation methodology. Sections 4 and 5 present

the experimental setup and results. Finally, Sections 6, 7, and 8 cover the discussion, conclusion, future work and ethical implications of this study.

2 Background and Research Gap

To understand how different learning rules map input data to output classifications, it is necessary to establish the mathematical and theoretical mechanisms behind the three algorithms evaluated in this study.

2.1 Backpropagation (The Baseline)

BP optimises neural networks by calculating the gradient of a loss function $\mathcal{L}$ with respect to each weight matrix $\mathbf{W}^l$ via the chain rule. For a given layer l , the weight update is defined as:

$$\Delta \mathbf{W}^l = -\eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^l} = -\eta \delta^l (\mathbf{a}^{l-1})^T, \quad (1)$$

where η is the learning rate, $\mathbf{a}^{l-1}$ is the forward activation of the previous layer, and δ^l represents the error propagated backwards from the subsequent layer. While highly effective at minimising error and creating linearly separable hidden representations, its reliance on global error signals, non-local weight updates, and high memory usage makes it the standard to beat in terms of accuracy, but the bottleneck in terms of hardware efficiency.

2.2 SoftHebb (Unsupervised Local Learning)

SoftHebb is a biologically inspired, unsupervised learning algorithm based on Hebbian plasticity. It implements a soft competition among neurons, where connections corresponding to stronger inputs are reinforced and weaker connections are depressed. The learning rule updates the synaptic weight w_{ij} between input x_i and neuron j according to the local post-synaptic activation y_j :

$$\Delta w_{ij} = \eta y_j (x_i - u_j w_{ij}), \quad (2)$$

where η is the learning rate and u_j is a local, post-synaptic normalisation variable that bounds the synaptic weights to prevent runaway reinforcement. The post-synaptic activation y_j is determined prior to this update via a soft winner-take-all mechanism. This mechanism utilizes a temperature factor t_{invert} to govern the competition, ensuring that while the strongest firing neuron receives the largest update, other neurons can still activate and learn proportionally. Because SoftHebb updates all neurons in proportion to their post-activation without requiring an external label, gradient computation, or feedback signal, it excels at finding the natural structural variance in a dataset. However, its purely unsupervised nature raises questions about how well it can disentangle data when natural visual clusters contradict assigned class labels.

2.3 PEPITA (Supervised Forward-Only Learning)

PEPITA (Present the Error to Perturb the Input To modulate Activity) bridges the gap between local learning and supervised feedback. It operates using two forward passes: a standard pass to generate an initial prediction $\mathbf{h}^L$, yielding an error $\mathbf{e} = \mathbf{h}^L - \mathbf{y}$. This is followed by a second "perturbed" pass where the error is projected back to the input space via a random projection matrix $\mathbf{B}$, meaning the perturbed input is $\mathbf{x}_{err} = \mathbf{x} + \mathbf{B}\mathbf{e}$. By comparing the local node activations of the first pass ($\mathbf{h}^l$) with the perturbed second pass ($\mathbf{h}_{err}^l$), PEPITA derives a localised weight update rule for the hidden layers:

$$\Delta \mathbf{W}^l = -\eta (\mathbf{h}^l - \mathbf{h}_{err}^l) (\mathbf{h}_{err}^{l-1})^T. \quad (3)$$

This difference-based formulation provides a task-directed learning signal while maintaining forward-only, localised weight updates that entirely bypass the standard backward pass.

2.4 Related Work and Research Gap

Biological and Hardware Motivations for BP-Free Learning. The push away from Backpropagation is motivated by the biological implausibility of the weight transport problem [1] and the computational bottleneck of update locking [2]. Recent comprehensive reviews [3] highlight how these constraints restrict the deployment of standard neural networks on energy-constrained and neuromorphic hardware. To address this, various localized and BP-free learning algorithms have been proposed. Hebbian learning [5] provides a biologically plausible foundation, leading to the development of unsupervised rules like SoftHebb [6]. Conversely, algorithms like PEPITA [7] attempt to maintain supervised, error-driven modulation while introducing forward-only updates. Furthermore, recent work by Guo et al. [11] demonstrates the viability of utilizing such stable, BP-free local learning rules specifically targeted towards physical learning systems.

Representational Analysis and Interpretability. While the algorithmic development of BP-free learning is accelerating, the tools used to evaluate them remain largely focused on raw classification accuracy. In contrast, the internal workings of standard gradient-based architectures have been extensively mapped. Foundational works in interpretability have established methods to visualize internal representations [12, 13] and understand feature extraction through saliency mapping [10]. Similarly, techniques like linear classifier probes [14] and Centered Kernel Alignment (CKA) [9] are widely used to measure the geometric similarity of hidden manifolds, such as comparing the distinct representational strategies of CNNs and Vision Transformers [4]. However, the application of these metrics to compare learning algorithms, especially purely localized and forward-only paradigms like SoftHebb and PEPITA remains a critical gap in the literature, which this study directly addresses.

3 Analytical Framework

To comprehensively analyse how BP, PEPITA, and SoftHebb structure their hidden spaces, we developed a multi-phase evaluation pipeline scaling from synthetic baselines to real-world visual datasets. The methodology is specifically designed to unpack the "black box" nature of Multi-Layer Perceptrons (MLPs) trained via Backpropagation, PEPITA, and SoftHebb, comparing their representational geometries, feature extraction capabilities, and computational hardware efficiencies.

3.1 Data Scaling Strategy

Our evaluation spans two distinct datasets, strategically chosen to progressively test the representational capacity and learning dynamics of the evaluated rules. The initial phase utilises a synthetic two-dimensional dataset explicitly generated to contrast the differences between supervised and unsupervised learning algorithms (Figure 1). In this small setting, the natural visual left-right clustering of the data directly contradicts the assigned class labels, which are split horizontally. This deliberate mismatch allows us to determine whether an unsupervised, structurally-driven algorithm like SoftHebb can overcome their reliance on structural variance.

Subsequently, we evaluate how these methods handle higher-dimensional image data by scaling the MLPs to the MNIST dataset. In this phase, the 28×28 pixel images are flattened into 784-dimensional input vectors, providing a more complex manifold for the networks to navigate and disentangle.

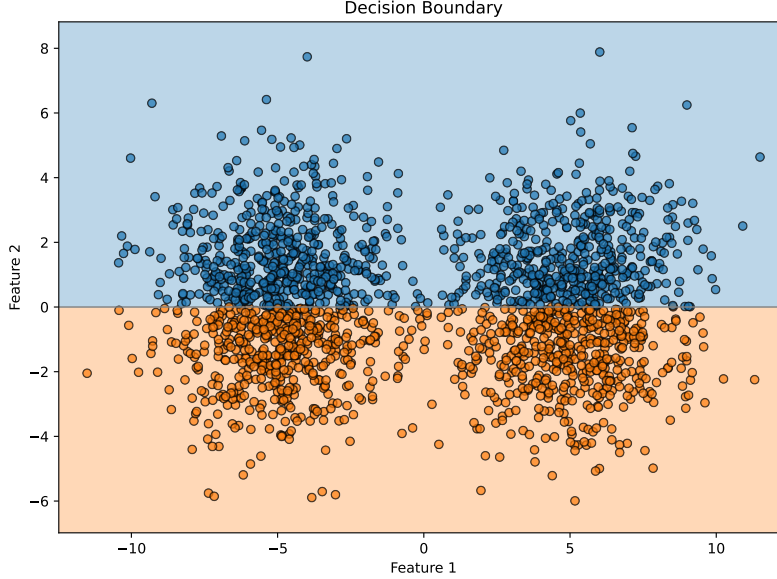


Figure 1: The synthetic data is made up of two clusters arranged left and right, but class labels are determined by whether points lie above or below the line $y = 0$.

3.2 Internal Representation and Similarity

While basic accuracy metrics are useful for confirming the overall behavioural capacity of an algorithm, they completely obscure the structural organisation of the internal hidden spaces. To peek inside these intermediate layers and properly analyse the representations they form, our methodology incorporates a suite of geometric and statistical metrics.

Linear Probing: We apply linear probing [14] by training simple, non-deep logistic regression classifiers directly on the frozen features extracted from intermediate layers. This technique measures how linearly separable the data becomes at specific depths within the network hierarchy.

Principal Component Analysis: To explicitly observe how each algorithm warps and disentangles contradictory inputs, we utilise PCA [8]. By compressing the high-dimensional hidden topological manifolds into a two-dimensional plane based on axes of maximum variance, PCA allows us to visually contrast the different clustering strategies and decision boundaries employed by each learning rule.

Linear Centered Kernel Alignment: Moving beyond visualisations, we require rigorous quantitative metrics to evaluate the similarity between the representational spaces generated by BP, PEPITA, and SoftHebb. To achieve this, we employ Linear Centered Kernel Alignment (CKA) [9]. CKA yields a scalar similarity index where scores approaching 1.0 indicate that the hidden spaces of two networks have converged to structurally equivalent mappings, despite being guided by fundamentally different learning rules. Lower scores, conversely, highlight highly distinct topological strategies.

Saliency Mapping: To observe the specific semantic features learned by each model, we utilize saliency mapping [10] to project intermediate visual features back to the pixel space. Rather than standard deconvolution approaches typically used for CNNs [13], we adapt this technique for MLPs by aggregating the responses of an entire hidden layer. Specifically, we calculate the gradient of the sum of all neuron activations in a given

layer l with respect to the input image vector x . This aggregate gradient forms a single saliency map for that depth, allowing us to visually inspect the specific patterns, textures, and structural pixels that are responsible for driving the layer’s overall activation across BP, PEPITA, and SoftHebb models.

3.3 Hardware Efficiency Benchmarking

The primary motivation for the transition towards alternative learning paradigms is the physical computing efficiency they theoretically afford over traditional backpropagation. To translate this theoretical advantage into measurable physical constraints, our final evaluation phase consists of an empirical hardware analysis. The benchmarking measures the total execution time, the average time required per training epoch, and the peak process memory usage across the three algorithms. To achieve completely isolated measurements and ensure a higher accuracy, we execute each benchmark in an entirely separate Python process. This strict isolation should eliminate any potential interference from background garbage collection, residual data structures, or memory caching between the different models.

4 Experimental Setup

To ensure a rigorous, and highly reproducible comparison across the distinct learning paradigms, we implemented and evaluated all neural network models under standardised configurations. Our experimental framework was structured to evaluate these systems across two primary tasks of increasing complexity. The first is a controlled, two-dimensional synthetic clustering task designed to highlight paradigm-level differences when label assignments contradict spatial structures. The second is a high-dimensional image classification task utilising the MNIST dataset, which tests the scaling capabilities of each algorithm when confronted with natural visual distributions.

4.1 Architectural Topology and Task Dimensions

We constructed Multi-Layer Perceptrons (MLPs) tailored specifically to the structural demands of each evaluation environment. As a whole, the MLP was defined with 3 layers: the first layer maps the input to the hidden dimension, the second layer is a fully connected layer maintaining the dimensionality of the hidden space, and the final layer, which acted as the classification layer, maps the hidden dimension to the output space. For the synthetic task, the input space was defined in two dimensions, the hidden dimension had 64 neuronal units, and the classification layer had two output classes. When scaling to the MNIST benchmark, we flattened the input 28×28 pixel grids into 784-dimensional vectors. Correspondingly, we expanded the hidden layers to contain 512 neuronal units, which projected to a final output layer representing the 10 target digit classes. Regarding the training duration, all models evaluated on the synthetic clustering task were trained for 50 epochs. When scaling to the primary MNIST representational evaluations, the models were trained for 100 epochs to ensure full convergence, with the exception of the isolated hardware benchmarking, which utilised a 50-epoch training run.

4.2 Hyperparameters and Optimiser

A comprehensive tabulation of all hyperparameters, learning rates, and structural parameters utilised across these experiments is provided in Appendix A. Furthermore, the core configurations and algorithmic baseline implementations are derived from the methodologies detailed in [3], which also provides direct links to the respective open-source codebases for each algorithm.

Baseline Backpropagation Our baseline Backpropagation models were optimised using standard Stochastic Gradient Descent (SGD) with a standard cross-entropy loss. SGD was selected to ensure a consistent optimisation baseline across all evaluated learning paradigms.

PEPITA Configuration PEPITA replaces the backward pass with an error-modulated second forward pass. We trained these models using momentum-based Stochastic Gradient Descent (SGD). A critical architectural component is the static, random projection matrix that projects the output error back into the input space. To prevent activation blow-up during perturbation, this matrix was initialised from a uniform distribution and scaled down by a factor of 0.05.

SoftHebb Configuration Operating unsupervised, SoftHebb relies on localised, post-activation-driven Hebbian updates, optimised via standard SGD. To govern the soft winner-take-all dynamics, we calibrated the temperature factor which ensures competitive learning between neurons without allowing individual ones to monopolise updates.

4.3 Empirical Hardware Benchmarking

To guarantee the absolute physical integrity of our computational benchmarks, we implemented meticulous environmental controls. Modern deep learning libraries like PyTorch dynamically manage system memory, often allocating shared caches, or delaying CPU garbage collection. Sequentially executing different training pipelines within a single Python runtime can therefore lead to cross-model contamination, where memory allocations or caching strategies from an earlier model artificially inflate or decrease the recorded footprint of a subsequent model. To avoid this variable, we isolated the execution of each individual algorithm in three different processes. The measurements were performed in the same setting as the MNIST tests, with the same hyperparameters but a 50-epoch training run. Averaging all epochs smooths out any random hardware variables to get a more stable result. By enforcing these strict boundaries, we ensured highly accurate measurements of execution time and peak process memory usage for Backpropagation, PEPITA, and SoftHebb. Furthermore, Appendix B details the specifications and components of the computer these tests were run on.

5 Experimental Results

5.1 Synthetic Dataset Results

In this dataset (Figure 1), the natural clustering of the data points along the horizontal axis directly contradicts the assigned classification labels, which are split vertically across the y-axis. The results (Figure 2) show that all three algorithms successfully map this contradiction. Backpropagation completely overrides the visual clusters to solve the task. Similarly, the supervised, error-modulated nature of PEPITA converges quickly, allowing it to accurately separate the space. SoftHebb, which relies strictly on local unsupervised Hebbian plasticity, also successfully disentangles the labels from the spatial clusters, but with the slowest convergence and noisiest accuracy graph. Ultimately, all three methods achieve a highly competitive final test accuracy of approximately 99%, with the performance ranking changing across different runs.

5.2 MNIST Representation Results

Scaling the algorithms to the higher-dimensional MNIST dataset furthermore demonstrates that Backpropagation remains the performance upper bound, achieving a test accuracy of 97.75%. In contrast, the forward-only PEPITA algorithm attains a test accuracy of 87.69%, and the entirely local SoftHebb model reaches 87.46% (Figure 3).

Despite this accuracy gap, linear probing on the frozen intermediate representations (Table 1) indicates that all three algorithms transform the complex input manifold into separable spaces starting from layer 1 (FC1). Notably, the linear probing accuracies for the intermediate layers of the forward-only models (94.42% for PEPITA and 94.39% for SoftHebb) significantly exceed their respective end-to-end network accuracies.

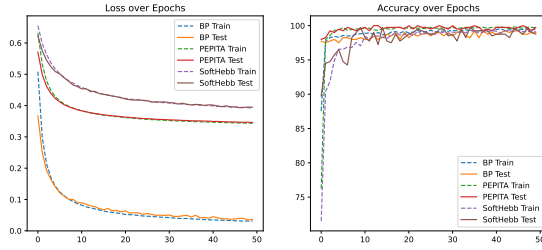


Figure 2: Accuracies of the three models on the synthetic dataset, with all methods achieving approximately 99% final accuracy on the testing set.

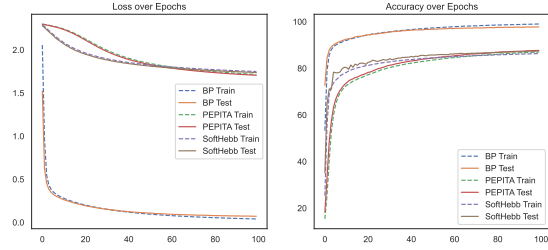


Figure 3: Accuracies and Loss functions of the three models on the MNIST dataset, with final accuracies on the testing set: BP 97.75%, PEPITA 87.69%, SoftHebb 87.46%.

Table 1: Linear Probing Test Accuracy by Method and Layer.

Method	Layer	Train Acc	Test Acc
BP	FC1	0.9996	0.9773
BP	FC2	0.9988	0.9785
PEP	FC1	0.9699	0.9442
PEP	FC2	0.9709	0.9449
SH	FC1	0.9687	0.9439
SH	FC2	0.9537	0.9365

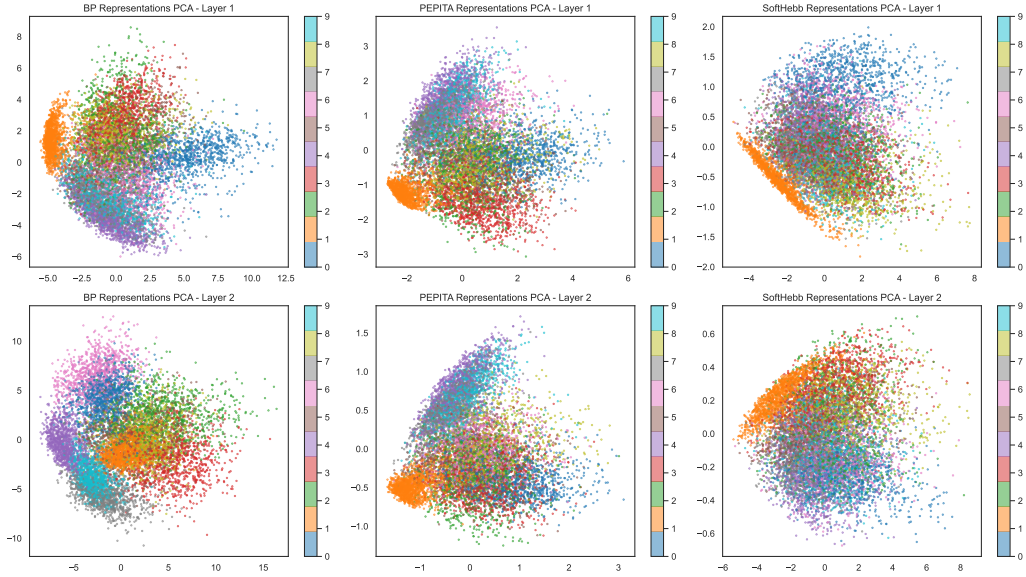


Figure 4: Principal Component Analysis (PCA) projections of the internal hidden manifolds (Layer 1 and Layer 2) showcasing the clustering strategies of BP, PEPITA, and SoftHebb.

Visualizing these high-dimensional hidden spaces via Principal Component Analysis (PCA) (Figure 4) confirms these geometric differences. Backpropagation effectively disentangles the data, mapping it into distinct, separated clusters by Layer 2, with little overlap between classes. PEPITA displays a structured separation of the digit classes, but with noticeably more visual overlap compared to BP. In contrast, SoftHebb’s manifolds appear highly entangled and radial in the 2D projection.

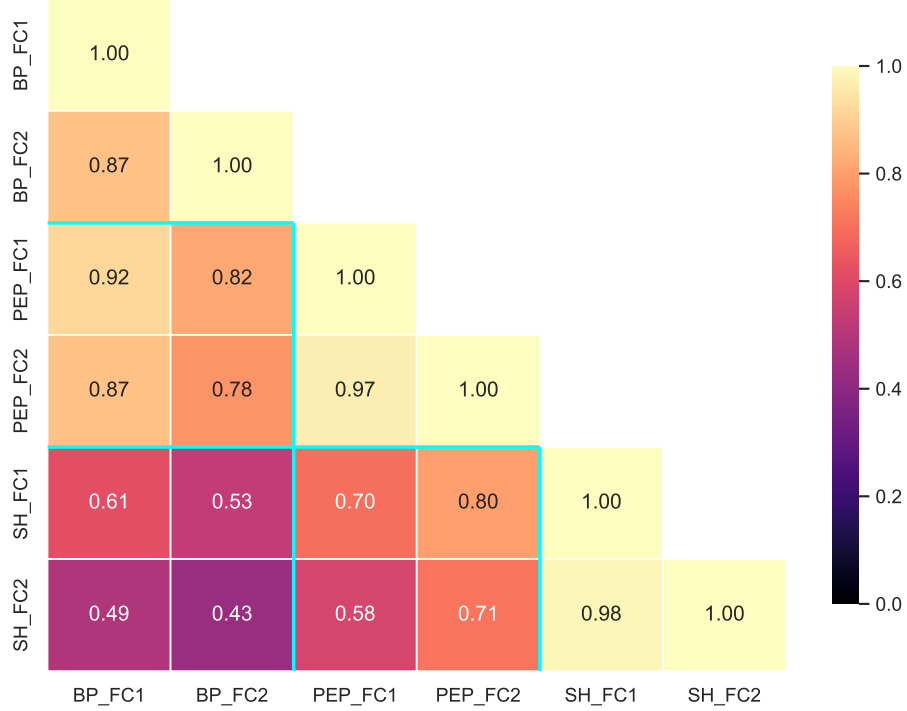


Figure 5: Cross-model and cross-layer Centered Kernel Alignment (CKA) heatmap comparing the structural similarities of representations generated by BP, PEPITA, and SoftHebb on the MNIST dataset.

A quantitative analysis using Centered Kernel Alignment (CKA) (Figure 5) reveals the structural similarities across the layers. In the first hidden layer (FC1), PEPITA shares a high CKA similarity with Backpropagation (0.92). Conversely, the internal representation of SoftHebb shares less geometric similarity with Backpropagation at this depth, yielding a CKA of 0.61. As data propagates to the second hidden layer (FC2), PEPITA’s representational space diverges further from BP (CKA dropping to 0.78) while increasing its alignment with SoftHebb (CKA of 0.80). SoftHebb remains distinct from BP in the deeper layer (CKA of 0.43).

An observation of the feature extraction strategies through saliency mapping (Figure 6) highlights how each network interprets the input. Projecting the activations for the digit ‘7’ back to the pixel space reveals that BP relies on highly focused, sparse activations concentrated on the edges of the digit and surrounding outlines. In contrast, PEPITA captures a broader, noisier contour of the digit’s overall shape in the first layer, refining this contour in the second layer by reducing background noise. SoftHebb’s activations follow the structural shape of the input digit the closest but retain a significant amount of background noise. This noisy representation correlates directly with SoftHebb’s final prediction distribution, which exhibits the highest uncertainty with the class ‘9’ compared to the near-absolute confidence of Backpropagation or the marginal uncertainty of PEPITA.

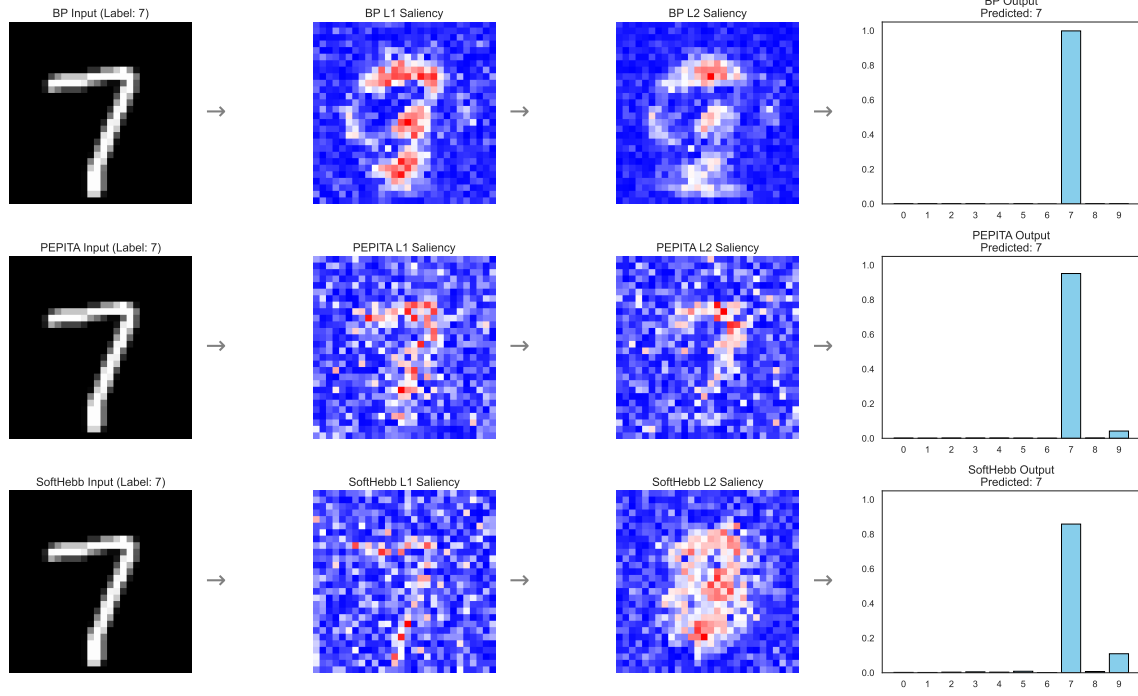


Figure 6: Layer-wise saliency maps and final output prediction distributions for the digit '7'. Backpropagation demonstrates sparse feature extraction with high confidence, while SoftHebb exhibits higher background noise and prediction uncertainty.

5.3 Hardware Efficiency Benchmarking

Forward-only learning rules are theoretically designed for computational efficiency, but benchmarking them on a standard consumer-grade ARM architecture reveals a hardware paradox. Table 2 summarizes the execution time and memory usage across the three methods over a 50-epoch training run on the MNIST dataset.

Table 2: Hardware efficiency and performance benchmarks over 50 training epochs on the MNIST dataset.

Algorithm	Total Time	Avg Time per Epoch	Peak Process Memory
Backpropagation	181.96 s	3.64 s	420.17 MB
PEPITA	249.95 s	5.00 s	466.30 MB
SoftHebb	230.77 s	4.62 s	449.64 MB

Standard Backpropagation proved highly efficient, requiring an average of 3.64 seconds per epoch with a peak process memory of 420.17 MB. SoftHebb demonstrated competitive computational speed, averaging 4.62 seconds per epoch, and achieved a peak footprint of 449.64 MB. In contrast, PEPITA introduced the highest computational overhead. The algorithm's requirement to execute a second, error-modulated forward

pass resulted in the slowest training time of 5.00 seconds per epoch and the highest peak memory usage at 466.30 MB.

It is important to note that the numbers presented in Table 2 reflect an example run. While the numerical values vary slightly across different executions, the relative hardware efficiency rankings remain established. Across all trials, Backpropagation consistently demonstrates the highest execution speed and lowest memory overhead, whereas PEPITA has the highest computational cost.

6 Discussion

The primary objective of this study was to unpack the "black box" of localized, forward-only learning paradigms by evaluating how their internal representations and computational efficiencies compare to standard Backpropagation. From a broad perspective, our evaluation reveals a distinct set of trade-offs: BP remains the upper bound for mathematically optimal class separation, while SoftHebb excels at finding natural, unsupervised structural variance, and PEPITA operates as a representational bridge between the two.

6.1 Representational Dynamics (SQ1 & SQ2)

To address our first sub-question (SQ1) regarding representational similarity, we evaluated the geometric topology of the hidden layers. The disparity between end-to-end network accuracy and internal linear separability is particularly revealing. As shown by the linear probing results, localized learning rules like PEPITA and SoftHebb cluster data based on structural variance as early as layer 1. Because the data is disentangled so early, a highly optimized external linear probe—which utilizes powerful mathematical solvers to find perfect convergence—can extract a significantly higher accuracy directly from these intermediate clusters. This exposes that the forward-only hidden layers successfully learn rich, robust representations, but the network is bottlenecked by its native final layer’s inability to separate the clusters as accurately as an external solver.

This high dimensional separability seems to contradict the PCA visualizations, which display SoftHebb’s manifolds as highly entangled and radial. However, this visual entanglement in 2D space is misleading. Because SoftHebb is strictly unsupervised, it organizes the 512-dimensional hidden space based on natural structural variance instead of assigned class labels. This visual overlap reflects how SoftHebb preserves a wide intra-class variance to capture structural diversity, whereas Backpropagation maximizes inter-class margins to force strict categorical separation.

The CKA metrics further demonstrate how PEPITA successfully fills the gap between supervised gradients and unsupervised variance. Notably, PEPITA exhibits high geometric similarity to BP in its shallowest layer (FC1), suggesting that the error-modulated input acts as a powerful, task-specific proxy for backpropagation at the earliest stages of the network. However, as data propagates deeper, PEPITA diverges from BP and becomes geometrically aligned with the unsupervised SoftHebb. This indicates that the impact of the error perturbation diffuses in deeper layers, allowing the network to rely more heavily on natural data clustering.

This geometric topology directly influences how the network extracts visual patterns, addressing our second sub-question (SQ2). Although SoftHebb and PEPITA operate under distinct paradigms—purely unsupervised local plasticity versus supervised, error-modulated feedback—the saliency maps reveal that their shared foundation in Hebbian-style weight updates leads to a similar extraction strategy. BP’s activations around the edges act as a discriminative mechanism to separate the target class from visually similar alternatives. Alternatively, both SoftHebb and PEPITA capture broader, noisier, and more structural contours of the input. Because SoftHebb operates entirely without global supervised feedback, it focuses on the raw shape of the input rather than discriminative boundaries, explaining its higher background noise and output prediction uncertainty.

6.2 Limitations and Scalability

A potential limitation when interpreting these representational dynamics is the reliance on the MNIST dataset. While MNIST provides a standard baseline, it remains a relatively constrained and simple visual distribution compared to natural images. However, the representational dynamics observed—specifically PEPITA’s bridging behavior and SoftHebb’s structural variance—were consistent across both the 2-dimensional synthetic clustering task and the 784-dimensional MNIST dataset. This multi-scale consistency suggests that these geometric behaviors are fundamental properties of these learning rules themselves, rather than dataset-specific artifacts. Consequently, while empirical verification on more complex distributions (such as CIFAR-10) is reserved for future work, we hypothesize that these core representational trade-offs will scale consistently.

6.3 The Hardware Efficiency Paradox (SQ3)

Finally, to address our third sub-question (SQ3) regarding computational performance, the empirical hardware analysis yielded counter-intuitive results. Theoretically, BP-free methods should be significantly faster and lighter by eliminating the computationally heavy backward pass and avoiding the update locking problem. However, PEPITA emerged as the slowest and most memory-intensive algorithm in our benchmarking. This discrepancy is an artifact of the software framework rather than a flaw in the algorithms themselves.

Backpropagation in PyTorch (`loss.backward()`) does not execute in pure Python; it triggers highly optimized, compiled C++ and CUDA routines that compute gradients simultaneously across the entire computational graph. In contrast, custom implementations of PEPITA and SoftHebb require calculating weight updates manually using for-loops across layers. Executing Python loops and performing manual matrix multiplications carries a massive computational overhead. Furthermore, PEPITA inherently requires two full forward passes per batch (a standard pass and an error-modulated pass), compounding the execution time.

The inflated memory usage of the BP-free models can similarly be attributed to framework limitations. PyTorch’s Autograd engine aggressively optimizes memory by freeing intermediate activation buffers the exact millisecond they are no longer needed for the backward pass. To implement PEPITA and SoftHebb manually, we utilized `register_forward_hook` to intercept and store layer activations in Python dictionaries, while intermediate weight updates were accumulated in lists. Because these massive tensors are held in standard Python data structures, the Python garbage collector cannot free them until the entire batch iteration concludes. This requirement to "remember" all weights and intermediate matrices simultaneously inflates the peak memory footprint.

Despite these severe implementation disadvantages, the final execution times remained remarkably comparable across the board. This narrow margin highlights the substantial theoretical performance boost these localized algorithms would provide if implemented natively. Because PEPITA and SoftHebb entirely bypass the update locking problem, they are parallelizable. If these algorithms were implemented as custom C++ PyTorch extensions or deployed directly onto neuromorphic substrates, their empirical efficiency would more accurately reflect their theoretical advantages over traditional Backpropagation.

7 Conclusions and Future Work

7.1 Conclusion

This thesis systematically evaluated the internal representations, learning mechanisms, and hardware efficiencies of Backpropagation, PEPITA, and SoftHebb. By unpacking the "black box" of these learning paradigms, we identified a clear set of trade-offs between biological plausibility, representational disentanglement, and classification accuracy.

Standard Backpropagation remains the performance upper bound, achieving the highest accuracy through rigid, highly disentangled, and mathematically optimal class separation. However, it requires global error signals and un-biological backward passes. SoftHebb demonstrated the power of purely local, unsupervised Hebbian plasticity, exhibiting remarkable memory efficiency and an innate ability to capture structural variance, though its lack of top-down feedback resulted in lower confidence and higher noise in complex classification tasks like MNIST. PEPITA emerged as a highly compelling intermediate solution. By utilizing a second, error-modulated forward pass, PEPITA successfully bridged the representational gap; it effectively mimics a supervised backward pass in shallow layers while shifting toward structural, unsupervised-like variance in deeper layers.

Furthermore, our hardware analysis uncovered a significant implementation paradox. While theoretical models suggest BP-free methods are inherently more efficient, our digital benchmarks showed PEPITA to be the most computationally expensive. We demonstrated that this is strictly a limitation of the high-level Python framework rather than an algorithmic flaw. Because forward-only and local update rules eliminate the update locking problem, they are innately parallelizable, making them ideal candidates for the next generation of energy-efficient computing.

7.2 Future Work

The findings of this comparative analysis open several promising avenues for future research:

Scaling to Complex Datasets: While MNIST provided a foundational benchmark, future research should evaluate these algorithms on larger, more complex visual distributions such as Fashion-MNIST or CIFAR-10. Analyzing whether PEPITA maintains its representational bridging behavior on highly complex, non-linear manifolds would be highly valuable.

Native Framework Implementations: To accurately measure the true computational efficiency of BP-free paradigms, future work should focus on implementing PEPITA and SoftHebb as low-level C++ or CUDA extensions within PyTorch. By eliminating Python loop overheads and optimizing memory allocation, researchers could directly compare their execution speeds against PyTorch’s highly optimized Autograd engine.

Hybrid Architectures: Given the representational similarities in early layers, future studies could explore hybrid network topologies. For instance, an architecture could leverage SoftHebb’s unsupervised, localized clustering for highly efficient, generalized early feature extraction, while employing PEPITA in the final layers to enforce task-specific classification. This layer-by-layer slicing would provide a fully localized, forward-only pipeline that captures the structural variance of Hebbian learning alongside the targeted accuracy of error-modulation.

Neuromorphic Hardware Deployment: The ultimate goal of BP-free algorithms is their deployment on physical neuromorphic substrates. Translating these trained models onto specialized hardware and measuring the real-world energy consumption would provide definitive proof of their sustainability benefits over traditional von Neumann architectures.

8 Responsible Research

8.1 Use of Artificial Intelligence Tools

During the preparation of this thesis, the Google Gemini large language model was utilized as a collaborative assistant to support the research process. Specifically, the model was employed to refine the grammar,

structure, and phrasing of the written sections to ensure optimal readability and academic tone. Furthermore, Gemini assisted in generating Python scripts for the complex data visualizations, debugging the algorithms, as well as formatting the $\text{\LaTeX}$ code for the document’s figures and tables. It is important to emphasize that while AI was utilized to enhance the presentation and formatting of the text, all core methodologies, algorithmic implementations, hardware analyses, and scientific conclusions remain the original work of the author.

8.2 Reproducibility and Open Source Code

Ensuring that the findings of this comparative analysis can be independently verified is a core principle of this research. To facilitate full transparency and reproducibility, the complete experimental pipeline—including model implementations, dataset generation scripts, and hardware profiling tools—has been open-sourced and is available at <https://github.com/seburebu111/analysis-hebbian-learning>.

The codebase adheres to strict reproducibility standards:

- **Standardized Frameworks:** All neural network models, geometric metrics (CKA, PCA), and hardware evaluations are implemented using widely adopted, open-source libraries, primarily PyTorch and scikit-learn.
- **Controlled Randomness:** For the synthetic baselines, the structural generation of the data clusters and the train/test splits are controlled using fixed random seeds to guarantee identical spatial constraints across independent trials.
- **Public Benchmarks:** The representational scaling experiments utilize the universally recognized MNIST dataset, ensuring that the accuracy trade-offs and hardware benchmarks can be reliably replicated by other researchers.

8.3 Ethical Considerations and Sustainability

A critical ethical consideration in modern artificial intelligence is the escalating environmental cost and immense energy consumption required to train deep learning models using standard Backpropagation (BP). Training state-of-the-art architectures relies on energy-intensive digital computing, contributing significantly to carbon emissions. By investigating and benchmarking BP-free learning alternatives like PEPITA and SoftHebb, this research directly contributes to the development of sustainable, energy-efficient AI paradigms. Because these localised learning rules eliminate the memory-heavy backward pass and avoid update locking, they offer a viable theoretical pathway for deploying neural networks on low-power, physical, and neuromorphic computing substrates.

Furthermore, the locality of Hebbian approaches like SoftHebb has been shown to naturally possess robustness against Gaussian noise and adversarial attacks (such as Projected Gradient Descent) without requiring explicit, computationally expensive defence mechanisms [15, 16]. Understanding how these algorithms internally structure their representations helps pave the way for deploying safer and more resilient AI models on edge devices.

References

- [1] Kendra S Burbank and Gabriel Kreiman. Depression-biased reverse plasticity rule is required for stable learning at top-down connections. *PLoS computational biology*, 8(3):e1002393, 2012.
- [2] Wojciech Marian Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *International Conference on Machine Learning*, pages 904–912. PMLR, 2017.

- [3] Rongguang Ye, Chenhao Ye, Chao Huang, Ming Tang, and Yunhao Liu. Beyond-backpropagation training: Methods, applications, and perspectives. *TechRxiv*, 2026. doi: 10.36227/techrxiv.176740426.63642005/v1.
- [4] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in neural information processing systems*, 34:12116–12128, 2021.
- [5] Michael A Arbib and James J Bonaiuto. *From neuron to cognition via computational neuroscience*. MIT Press, 2016.
- [6] Timoleon Moraitis, Dmitry Toichkin, Adrien Journé, Yansong Chua, and Qinghai Guo. Softhebb: Bayesian inference in unsupervised hebbian soft winner-take-all networks. *Neuromorphic computing and engineering*, 2(4):044017, 2022.
- [7] G Dellaferrera and G Kreiman. Error-driven input modulation: Solving the credit assignment problem without a backward pass. *arXiv 2022. arXiv preprint arXiv:2201.11665*.
- [8] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.
- [9] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMIR, 2019.
- [10] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [11] Yaqi Guo, Fabian Braun, Bastiaan Ketelaar, Stephanie Tan, Richard Norte, and Siddhant Kumar. Local learning for stable backpropagation-free neural network training towards physical learning. *arXiv preprint arXiv:2603.24790*, 2026.
- [12] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.
- [13] Wei Yu, Kuiyuan Yang, Yalong Bai, Hongxun Yao, and Yong Rui. Visualizing and comparing convolutional neural networks. *arXiv preprint arXiv:1412.6631*, 2014.
- [14] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.
- [15] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

A Model Configurations and Experimental Setup

A.1 Synthetic Dataset

The synthetic dataset consisted of 2,000 total samples, split into 80% training and 20% testing sets, and processed with a batch size of 32. The visual clusters were generated with a standard deviation of 2, centered at $[-5, 0]$ and $[5, 0]$. A fixed random seed of 42 was used for synthetic data generation.

Table 3: Model Architectures and Hyperparameters (Synthetic Task)

Parameter	Backpropagation	PEPITA	SoftHebb
Input Dimension	2	2	2
Hidden Dimension	64	64	64
Output Dimension	2	2	2
Optimizer	SGD	SGD	SGD
Learning Rate (η)	0.01	0.01	0.01
Algorithm Specifics	Standard Auto-diff	$\mathbf{B} \sim \mathcal{U}(-sd, sd) \times 0.05$ where $sd = \sqrt{6/2}$	$t_{invert} = 12.0$

A.2 MNIST Dataset

Imported with the `torchvision.datasets.MNIST` library function, the publicly available MNIST dataset consisted of 60,000 training and 10,000 testing samples. The 28×28 pixel images were flattened into 784-dimensional input vectors and processed with a batch size of 128.

Table 4: Model Architectures and Hyperparameters (MNIST Task)

Parameter	Backpropagation	PEPITA	SoftHebb
Input Dimension	784	784	784
Hidden Dimension	512	512	512
Output Dimension	10	10	10
Optimizer	SGD	SGD	SGD
Learning Rate (η)	0.01	0.01	0.03
Algorithm Specifics	Standard Auto-diff	$\mathbf{B} \sim \mathcal{U}(-sd, sd) \times 0.05$ where $sd = \sqrt{1/784}$	$t_{invert} = 12.0$

B Computer Specifications

Table 5: Hardware Specifications for Experimental Benchmarking.

Component	Specification
Model	MacBook Pro (Mac16,1)
Processor (Chip)	Apple M4 (10 cores: 4 performance, 6 efficiency)
Memory (RAM)	24 GB Unified Memory
Graphics (GPU)	Apple M4 GPU (10 cores)
Storage	1 TB APPLE SSD AP1024Z
Operating System	macOS Sequoia 15.6

The hardware environment used for the empirical benchmarking of the Backpropagation, PEPITA, and SoftHebb algorithms is detailed in Table 5. This configuration was used for the execution of the MNIST evaluation experiments to ensure transparency of the results.