Neural Caching: Improving Longevity of Smart IoT Devices running Artificial Neural Networks

Christian Sallinger
christian.sallinger@tuwien.ac.at
TU Wien
Vienna, Austria

Christian Stippel
christian.stippel@tuwien.ac.at
TU Wien
Vienna, Austria

Elias Panner
elias.panner@student.tuwien.ac.at
TU Wien
Vienna, Austria

Paul Poschenreither
paul.poschenreither@fraunhofer.at
Fraunhofer Austria Research GmbH
Vienna, Austria

Ralph Hoch
ralph.hoch@vactory.at
Digital Factory Vorarlberg GmbH
Dornbirn, Austria

Benjamin Schwendinger
benjamin.schwendinger@fraunhofer.at
Fraunhofer Austria Research GmbH
Vienna, Austria

Abstract

Resource constraints and hardware aging make long-term neural inference on embedded and IoT devices increasingly challenging. We present *Neural Caching*, a lightweight inference mechanism that exploits the piecewise-linear structure of Artificial Neural Networks (ANNs) with piecewise linear activation functions to accelerate evaluation without retraining or model compression. Our approach, *Neural Caching*, associates each locally linear region of an ANN with a cached affine mapping and reuses it for subsequent inputs falling within the same or nearby activation region. This enables full predictions to be computed via a single matrix multiplication instead of potential multiple matrix multiplications. Experiments on four human-activity recognition datasets demonstrate up to an order-of-magnitude reduction in inference time while preserving classification performance metrics within ±0.1 % of baseline accuracy. By lowering computational load and power draw, neural caching extends device lifetime and supports sustainable, long-term deployment of machine-learning models in real-world IoT environments.

CCS Concepts

• **Computer systems organization** → **Neural networks**; • **Theory of computation** → *Caching and paging algorithms.*

Keywords

artificial neural networks, inference-time caching, hardware longevity, time-series classification

1 Introduction

The rapid proliferation of IoT devices has fundamentally changed how we sense, collect, and act upon environmental data. Millions of connected sensors now operate continuously in smart cities, industrial monitoring, agriculture, and healthcare, often in resource-constrained and physically exposed environments. A persistent challenge in these systems is *longevity*: as hardware ages, sensors and embedded processors experience gradual degradation—battery capacities decline, compute throughput diminishes due to transistor wear and thermal stress [Kraak et al. 2018; Zaidi et al. 2024], and replacement is often infeasible. Sustaining reliable and efficient machine-learning inference under such conditions remains an open problem.

Neural networks have become a dominant approach for analyzing complex sensor data, owing to their ability to capture nonlinear dependencies and generalize across heterogeneous and noisy environments. However, their deployment on IoT devices remains limited by computational and energy constraints: deep models require substantial processing power and memory, accelerating battery depletion and reducing device lifespan. Common model-compression techniques—such as pruning, quantization, and knowledge distillation—help mitigate these costs, yet they often require retraining and may introduce slight losses in predictive accuracy. Moreover, most of these approaches assume static hardware performance and do not adapt to gradual degradation or varying operating conditions over time.

Recent theoretical work has shown that neural networks with piecewise-linear activation functions (such as Rectified Linear Unit (ReLU) or hard tanh) partition their input space into exponentially many linear regions [Montúfar et al. 2014]. Within each region, the network behaves as a single affine transformation. This structure opens the door to alternative forms of acceleration that exploit redundancy in repeated inference: if multiple inputs fall into the same linear region, their outputs can be computed by a cached linear mapping instead of full forward propagation.

In this work, we build upon this insight and propose an inference-time caching mechanism tailored for IoT time-series models. While caching has been explored in convolutional and vision networks—e.g., through feature reuse across video frames or linear-region enumeration for small Multilayer Perceptrons (MLPs) [Joyce and Verschelde 2024, 2025]—it has not, to our knowledge, been applied to continuous time-series data on embedded IoT platforms. Our method exploits the local linearity of ReLU networks to cache region-wise affine transformations encountered during normal operation. Subsequent inferences that fall into known regions are executed by lightweight matrix multiplications, yielding substantial compute and energy savings.

This approach directly addresses the longevity issue of IoT devices. As compute resources degrade and batteries age, the algorithm maintains acceptable latency through increasing cache hit rates, effectively compensating for lost performance. In addition, by reducing active computation and the number of recharge cycles, it

extends sensor lifetime and supports sustainable, long-term operation without sacrificing model capacity. Larger, more generalizable models—previously impractical on constrained hardware—can thus be deployed efficiently using our region-based caching scheme.

Our main contributions are as follows:

- We formulate a region-based caching algorithm for piecewise-linear neural networks and adapt it for time-series inference on embedded IoT devices.
- We demonstrate that caching affine mappings across linear regions yields consistent inference-time speedups as hardware performance degrades.
- We show that our approach preserves model accuracy while significantly reducing used computation power, thereby prolonging device longevity.

The remainder of this paper is organized as follows. Section 2 reviews related work on neural network acceleration and caching. Section 3 outlines the theoretical background of ReLU linear regions. Section 4 introduces the proposed Hierarchical Hypersphere Caching algorithm, followed by experiments and results in Section 5. Section 6 shows the limitations of our work. Section 7 concludes the paper and discusses future research directions.

2 Related Work

Neural-network acceleration without retraining has been explored from several complementary directions, including (i) exploiting piecewise-linear structure, (ii) inference-time caching, (iii) conditional computation, and (iv) lookup-table acceleration.

Piecewise-Linear and Region-Based Methods. ReLU networks partition input space into linear regions, each defining an affine mapping. Early *locally weighted learning* approaches such as locally weighted regression and locally weighted projection regression [Atkeson et al. 1997] approximated nonlinear functions through local linear models, effectively forming a piecewise-linear surface but requiring large memory and neighbor search—impractical for embedded devices. Mixture-of-Experts (MoE) networks [Shazeer et al. 2017] achieve conditional computation by activating only a subset of sub-networks per input, but increase model size and training complexity. Joyce and Verschelde [Joyce and Verschelde 2025] formally characterized how each activation pattern in a ReLU network corresponds to a unique affine mapping, providing the theoretical basis for region caching. Their work was primarily analytical and demonstrated on small multilayer perceptrons. Building on this principle, we extend the concept to time-series inference on IoT hardware, where activation patterns recur naturally. Unlike prior theoretical studies, we address limited cache memory, hardware degradation, and continuous deployment, thereby enabling longevity without retraining.

Inference-Time Caching and Feature Reuse. Balasubramanian et al. [Balasubramanian et al. 2021] introduced *GATI*, which learns to predict layer outputs from cached representations, achieving up to 7.7× latency reduction in cloud settings. However, as it relies on auxiliary models and continuous retraining, it is unsuitable for low-power IoT nodes. In contrast, our cache stores numerically exact affine mappings from previous activations, requiring no additional learning. Feature reuse has also been explored in vision. DeepCache [Xu et al. 2018] caches convolutional feature maps across similar video frames, achieving ∼18% speedup and 20% energy savings on mobile devices. These approaches exploit spatial or temporal locality but depend on heuristic frame matching. Our region-level cache instead operates on the network's intrinsic linear structure and guarantees exact reuse for repeated activation patterns.

Conditional Computation. Early-exit networks and MoE models achieve conditional computation by activating only part of the network—either exiting early or selecting a subset of experts per input—trading slight accuracy loss for speed [Addad et al. 2023; Bolukbasi et al. 2017; Shazeer et al. 2017; Teerapittayanon et al. 2016]. While effective, both require architectural changes or retraining. In contrast, our approach performs *exact computation reuse* within an unmodified pre-trained model and can complement such dynamic-inference methods.

Lookup-Table Accelerators. Lookup-table (LUT) methods replace arithmetic with indexed lookups. *MADDNESS* [Blalock and Guttag 2021] precomputes subspace dot products to approximate matrix multiplications, reducing compute at the cost of minor accuracy loss. *LUT-NA* [Sen et al. 2024] performs exact digital inference via low-precision LUT operations, achieving up to 3.3× lower energy and 29× smaller area without retraining. These hardware optimizations complement our method: LUTs accelerate every operation, while caching avoids redundant ones. Both align with inference-time, model-agnostic efficiency gains.

Overall, prior work accelerates inference through data locality, architectural sparsity, or hardware substitution. Our method specifically applies linear-region caching to time-series neural inference on IoT devices—reusing exact affine mappings across repeated activation patterns to reduce compute and energy consumption under hardware aging, thereby extending device longevity without retraining or model modification.

3 Background

This section introduces the notation and theoretical concepts underlying ReLU-based feedforward neural networks used in this work.

Network structure. We consider a neural network $f_\theta : \Omega \to \mathbb{R}^C$ with trainable parameters θ. The domain $\Omega \subset \mathbb{R}^D$ denotes the input space of dimension D, and the output $\mathbb{R}^C$ corresponds to the unnormalized class logits. Each hidden layer $l \in \{1, \ldots, L\}$ consists of d_l neurons, parameterized by a weight matrix $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and bias vector $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$. Given an input $\mathbf{x} \in \Omega$, the forward pass proceeds recursively as

$$\mathbf{p}^{(l)} = W^{(l)}\mathbf{q}^{(l-1)} + \mathbf{b}^{(l)}, \quad \mathbf{q}^{(l)} = \sigma\left(\mathbf{p}^{(l)}\right), \tag{1}$$

where $\mathbf{q}^{(0)} := \mathbf{x}$ denotes the input layer. The final layer is linear:

$$f_\theta(\mathbf{x}) = W^{(L+1)}\mathbf{q}^{(L)} + \mathbf{b}^{(L+1)}. \tag{2}$$

The resulting vector $f_\theta(\mathbf{x})$ contains one logit per class, typically transformed by a softmax function for probabilistic classification.

https://doi.org/10.34749/3061-1008.2025.1

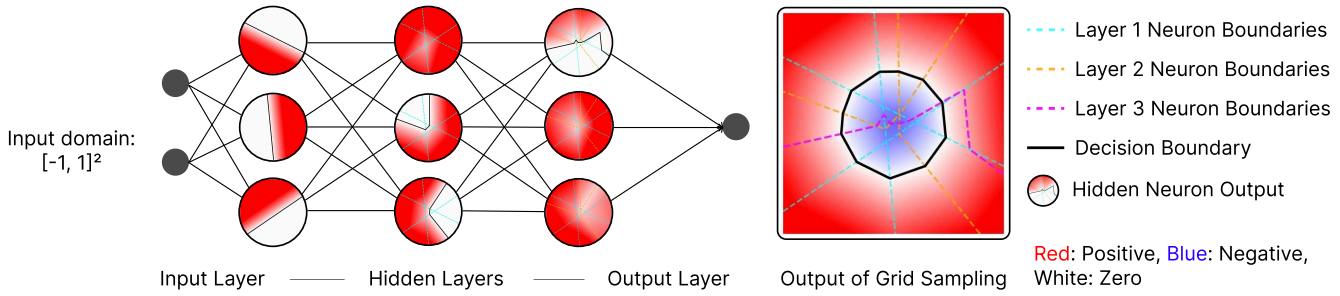

Figure 1: Linear regions in a ReLU MLP trained on a synthetic signed distance dataset of a circle. Dashed lines denote distinct affine regions of the ReLU network. Red indicates positive, blue negative values and white zero.

Piecewise linearity. The ReLU activation [Nair and Hinton 2010] is defined as

$$\sigma(x) = \max(0, x), \tag{3}$$

and is linear on each side of its non-differentiable point at zero. Because every layer in Eqs. (1)–(2) is an affine map followed by a ReLU, their composition is also affine on local subsets of the input space. Thus, any feedforward ReLU network (in this case f_θ) represents a continuous piecewise-linear function [Arora et al. 2018; Hanin and Rolnick 2019]. Each neuron defines a hyperplane

$$\{\mathbf{x} \in \mathbb{R}^D \mid \mathbf{p}_i^{(l)}(\mathbf{x}) = 0\}, \tag{4}$$

that partitions the space into two half-spaces where the neuron is *active* ($\mathbf{p} > 0$) or *inactive* ($\mathbf{p} \leq 0$). The pattern of active neurons determines a *region of linearity*, within which the network behaves as a single affine mapping [Poole et al. 2016; Raghu et al. 2017].

Partitioning into linear regions. The intersection of all neuron hyperplanes across layers subdivides Ω into convex polytopes or *linear regions* [Hanin and Rolnick 2019; Montúfar et al. 2014]. Each region corresponds to a unique binary activation pattern across neurons, and the network is affine inside that region:

$$\widetilde{f}(\mathbf{x}) = \widetilde{W}\,\mathbf{x} + \widetilde{\mathbf{b}}. \tag{5}$$

Here, $\widetilde{W}$ and $\widetilde{\mathbf{b}}$ are obtained by collapsing all active submatrices of $W^{(l)}$ and corresponding biases through the network hierarchy. The affine form in Eq. (5) remains valid within the region C and changes only when the input crosses a neuron boundary, activating or deactivating a unit. Figure 1 shows the regions for a two-dimensional example trained on a synthetic dataset representing a signed distance function of a circle. The input domain is sampled on a grid; each circle represents the output of the neuron after applying ReLU. Dashed lines indicate the neuron boundaries (zero-level set) that separate the piecewise linear parts.

Geometrically, each hidden neuron introduces a half-space constraint, and deeper layers iteratively refine the subdivision of Ω. The number of regions grows combinatorially with depth: a deeper ReLU network can carve exponentially more regions than a shallow one with a similar parameter count [Poole et al. 2016; Raghu et al. 2017]. Montúfar *et al.* [Montúfar et al. 2014] established lower bounds on this growth, showing that an L-layer network with n neurons per layer can realize on the order of $\Omega((n/n_0)^{(L-1)n_0} n^{n_0})$ distinct regions, where n_0 is input dimension. Telgarsky [Telgarsky 2016] further demonstrated that depth yields exponentially more expressive power: there exist functions representable by a depth-$\Theta(k^3)$ network that cannot be approximated by any network of depth $O(k)$ unless it has exponentially many neurons. These findings formalize the intuition that deeper networks can form more complex decision boundaries through hierarchical region composition. For classification, transitions between classes occur at the boundaries where two logits are equal, forming a piecewise-linear approximation of the underlying decision manifold [Hanin and Rolnick 2019].

4 Hierarchical Hypersphere Caching

The piecewise-linear structure of ReLU networks partitions the input space into a collection of linear regions where the network behaves as an affine function [Arora et al. 2018; Hanin and Rolnick 2019; Montúfar et al. 2014]. This enables reusing previously computed linearizations for any new input that falls into a region encountered before [Joyce and Verschelde 2025]. Each time a ReLU MLP is evaluated at a point $\mathbf{x}_0$, the pattern of active and inactive neurons defines a local *linear cell* (a convex polytope) within which the network acts as a single affine mapping $\widetilde{f}(\mathbf{x}) = \widetilde{W}\mathbf{x} + \widetilde{\mathbf{b}}$. Although a deep network can, in theory, partition its domain into an exponential number of such regions [Montúfar et al. 2014], empirical studies show that real ReLU networks realize far fewer distinct activation patterns [Hanin and Rolnick 2019]. This redundancy suggests that large portions of the input space share the same affine transformation, creating opportunities for computational reuse. Joyce and Verschelde [Joyce and Verschelde 2025] formalized this insight by showing that each unique ReLU activation pattern corresponds to a distinct affine mapping that can be memoized for faster inference.

Hypersphere caching. Our proposed *Hypersphere Cache* associates each cached linearization with the largest *inscribed hypersphere* centered at $\mathbf{x}_0$ that remains fully inside the corresponding linear region. Its radius is determined by the minimum Euclidean distance to any neuron activation boundary. Geometrically, this sphere defines the maximal region around $\mathbf{x}_0$ in which all ReLU activations remain constant. Hence, if a future query point $\mathbf{x}$ lies within an existing sphere, the stored affine mapping is reused directly (a *cache hit*); otherwise, a new linearization is computed and

https://doi.org/10.34749/3061-1008.2025.1

stored (a *cache miss*), analogous to region-based caching [Joyce and Verschelde 2025] and feature reuse approaches such as GATI [Balasubramanian et al. 2021] or LUT-based inference methods [Blalock and Guttag 2021; Sen et al. 2024].

Cache representation. The cache maintains tuples of the form $(\mathbf{x}_0, r, W_{\text{final}}, \mathbf{b}_{\text{final}})$, where $\mathbf{x}_0 \in \mathbb{R}^D$ is the hypersphere center, $r \in \mathbb{R}^+$ its radius, and $W_{\text{final}} \in \mathbb{R}^{C\times D}$, $\mathbf{b}_{\text{final}} \in \mathbb{R}^C$ define the affine mapping. Each entry satisfies $f_\theta(\mathbf{x}) = W_{\text{final}}\mathbf{x}+\mathbf{b}_{\text{final}}$ for all $\mathbf{x}$ within the hypersphere. When a new query arrives, the cache searches for a containing sphere and either reuses the stored linearization or computes a new one.

Hierarchical lookup strategy. To achieve fast lookup times, we use a hierarchical three-stage search exploiting temporal and spatial locality:

(1) **Last-hit check (O(1))**: The previously used entry `last_idx` is checked first, as consecutive inputs often lie in the same region. If $\|\mathbf{x} - \mathbf{x}_0\|_2 < r$, the cached linearization is reused immediately.
(2) **Recent insertions (O(k))**: Newly added spheres not yet in the spatial index are searched next, enabling immediate reusability.
(3) **Spatial index (O($\log n$))**: Older cached entries are organized in a ball tree using Euclidean distance. The depth of this tree grows logarithmically. Given the maximum stored radius $r_{\max}$, the index retrieves all hyperspheres with centers within $r_{\max}$ of $\mathbf{x}$. Only a small subset of candidates are then distance-checked for containment.

This hierarchical structure combines constant-time reuse in common cases with logarithmic-time spatial search in the general case, yielding fast average query performance even as the cache grows.

Cache updates and rebuilds. On a cache miss, the network performs a full forward pass to compute a new linearization at $\mathbf{x}$. The valid radius is determined as $r = \min_{l,i} |r_{l,i}|$, ensuring conservative containment within the linear region. The new entry is appended to the cache and integrated into the ball tree once the number of pending insertions exceeds a threshold, amortizing the $O(n \log n)$ cost of index maintenance.

Illustration. Fig. 2 visualizes a two-dimensional example. Cyan, orange, and magenta lines denote neuron boundaries from three hidden layers, subdividing the input into convex linear cells. Dotted circles indicate cached hyperspheres. Orange circles mark new linearizations (cache misses), while blue squares indicate reuse events (cache hits). Larger hyperspheres appear in stable regions far from activation boundaries, while smaller spheres cluster near decision boundaries where neuron states change frequently. This behavior mirrors results from activation pattern analyses [Hanin and Rolnick 2019; Raghu et al. 2017] and illustrates how the cache adapts to local model smoothness.

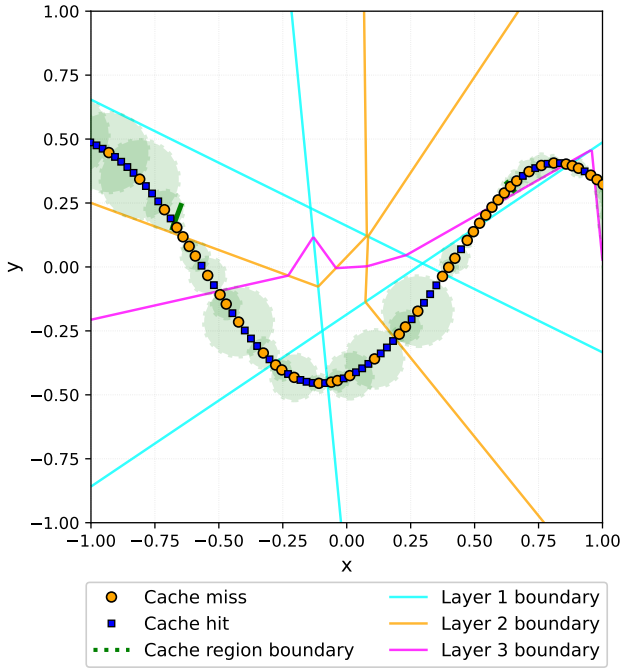

Figure 2: Caching visualization of a model with 2 input neurons.

5 Evaluation & Results

5.1 Datasets

We evaluate our method on four publicly available human activity recognition (HAR) datasets: *WISDM*, *MHEALTH*, *PAMAP2*, and *UCI-HAR*. A summary of their key characteristics is shown in Table 1.

WISDM. The WISDM dataset [Kwapisz et al. 2011] contains smartphone and smartwatch motion-sensor recordings (accelerometer and gyroscope) from 51 subjects performing everyday activities. We use the raw smartphone accelerometer signals sampled at 20 Hz and focus on non-hand-oriented activities (walking, jogging, climbing stairs, and standing). Sensor data are segmented using 2.5 s sliding windows (50 readings per window) with 50 % overlap. For each axis, five statistical features (mean, standard deviation, minimum, maximum, median) are extracted to form 15-dimensional feature vectors.

MHEALTH. The MHEALTH dataset [Baños et al. 2014] includes body motion and vital sign recordings from ten volunteers equipped with three sensors (chest, right wrist, left ankle), each providing 23 readings at 50 Hz. We consider six activities (standing, sitting, walking, climbing stairs, jogging, and running). Windows of 2 s (100 readings) with 50 % overlap are used, and features are extracted analogously to WISDM.

Table 1: Summary of datasets used for evaluation.

Dataset	Input Features	Activities	Samples
WISDM	15	5	53,475
MHEALTH	115	7	4,290
PAMAP2	124	6	24,600
UCI HAR	561	6	10,299

https://doi.org/10.34749/3061-1008.2025.1

PAMAP2. The Physical Activity Monitoring (PAMAP2) dataset contains recordings from nine subjects wearing three inertial measurement units (IMUs) and a heart-rate monitor [Reiss and Stricker 2012]. The IMUs sample at 100 Hz and the heart-rate sensor at 9 Hz. We focus on lying, sitting, standing, walking, and ascending/descending stairs. Feature extraction follows the same procedure using 1 s windows (100 readings) with 50 % overlap.

UCI-HAR. The UCI-HAR dataset [Anguita et al. 2013] comprises accelerometer and gyroscope signals from 30 volunteers performing six activities (walking, walking upstairs/downstairs, sitting, standing, and lying). The smartphone-mounted sensors sample at 50 Hz. Preprocessed windows of 2.56 s (128 readings) with 50 % overlap are provided, each represented by 561 pre-extracted features.

Train–test splits. For UCI-HAR, the predefined subject split is used. The remaining datasets are partitioned by subject: approximately 20 % of subjects are held out for testing, while the others form the training set. Features are extracted per subject, and test data preserve temporal continuity within each subject—crucial for evaluating our caching algorithm under realistic, continuous trajectories.

5.2 Models

We employ MLP architectures with ReLU activations for IoT time-series classification. The choice of ReLU is intentional: as a piecewise-linear activation, it partitions the input space into exponentially many linear regions [Raghu et al. 2017], which directly enables our region-based caching optimization.

All models use ReLU activations in the hidden layers and a linear output layer that produces logits for softmax-based multi-class activity classification. The input dimensionality corresponds to the feature size of each dataset: WISDM (15), MHEALTH (115), PAMAP2 (124), and UCI-HAR (561).

Training Configuration. Models are trained using the Adam optimizer [Kingma and Ba 2017], for 100 epochs, a learning rate of 0.001, batch size 64, and cross-entropy loss. The model with the lowest validation loss is saved.

Hyperparameter Optimization. A grid search varying network width (8–512 hidden units) and depth (1–5 layers) identifies optimal architectures for each dataset. Figure 3 shows validation accuracy as a function of hidden-layer width, averaged across depths.
The best configurations are: WISDM (256 units), MHEALTH (64 units), PAMAP2 (256 units), and UCI_HAR (128 units). Datasets requiring fewer layers compensate with wider hidden dimensions.

Figure 4 presents validation accuracy versus network depth, averaged over all widths.
The selected architectures are: WISDM (3 layers), MHEALTH (4 layers), PAMAP2 (1 layer), and UCI_HAR (1 layer). Simpler datasets achieve peak accuracy with shallow networks, whereas WISDM and MHEALTH benefit from deeper architectures capturing more complex feature interactions.

5.3 Caching

We evaluate the proposed hypersphere caching mechanism across four HAR datasets. The cache exploits the local linearity of ReLU networks by storing affine approximations valid within hyperspherical regions around previously seen inputs, as defined in Section 4. Evaluation is performed in an online inference setting over continuous test sequences, where each incoming sample either reuses a cached linearization or triggers a new one.

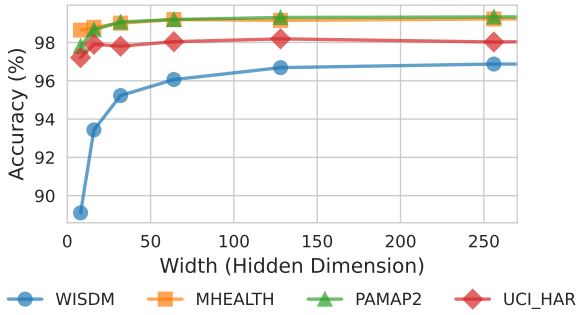

Figure 3: Accuracy vs. hidden dimension width, averaged across depths.

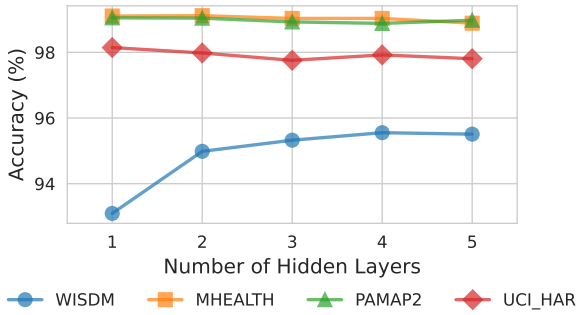

Figure 4: Accuracy vs. number of hidden layers, averaged across widths.

Cache construction and quantile-based radius selection. For each cached point, we compute the distance to the nearest neuron decision boundary at each layer, as defined in Section 4, i.e., the minimum Euclidean distance to any activation boundary derived from the layer's linearization at that point. These distances define the maximum radius within which the linearization remains exact. We store these radii and sort them to create a distribution, from which we select cache radii based on quantiles (10th to 40th percentile). This quantile-based selection allows us to trade off between cache hit rate and approximation quality.

Cache hit rates across datasets. Figure 5 shows the cache hit rate as a function of the selected radius quantile. WISDM exhibits the highest cache utilization, with hit rates increasing from 34% at the 10th percentile to 55% at the 40th percentile.
This indicates that over half of incoming samples fall within previously cached regions when using larger radii. MHEALTH and

https://doi.org/10.34749/3061-1008.2025.1

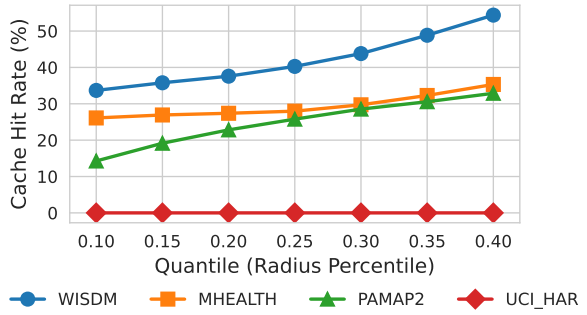

Figure 5: Cache hit rate vs. radius quantile. Higher quantiles correspond to larger hypersphere radii, increasing the likelihood of cache reuse.

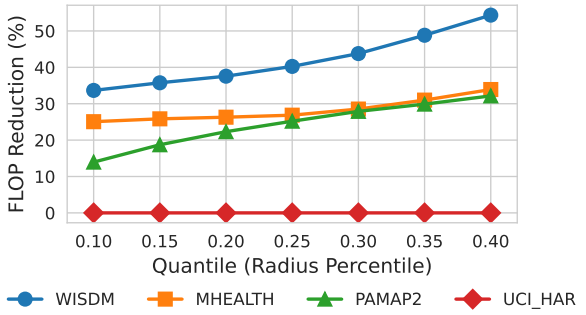

Figure 7: FLOP reduction vs. radius quantile. Values represent computational savings from using cached linear approximations instead of full forward passes.

PAMAP2 show moderate cache effectiveness, with hit rates climbing from 27% and 15% respectively at the smallest radii to approximately 35% and 33% at the largest. Notably, UCI-HAR shows negligible cache reuse across all quantiles, suggesting its input patterns do not revisit similar regions during inference. The cache hit rate correlates strongly with the input dimensions. Table 1 shows that WISDM uses 15 input dimensions, whereas UCI HAR has 561 input dimensions.

Preservation of model accuracy. Expanding the cache radius to enable limited extrapolative reuse does not degrade classification accuracy. Figure 6 shows that accuracy remains effectively constant across all quantiles:

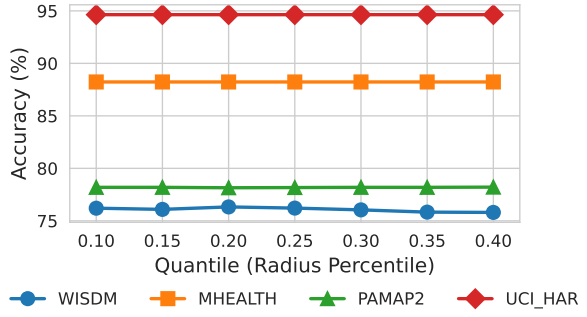

Figure 6: Classification accuracy vs. radius quantile. Accuracy remains stable across all quantile selections, indicating no degradation from extrapolative reuse.

UCI-HAR 95 %, MHEALTH 88 %, PAMAP2 78 %, and WISDM 76 %, all within ±0.1 percentage points. This stability confirms that cached affine approximations remain valid even slightly beyond their theoretical region boundaries.

Computational savings. Figure 7 quantifies the FLOP reduction achieved through caching, comparing the computational cost of using cached linear models versus full forward passes.
Reduction patterns mirror hit rates: WISDM achieves 34-55% fewer FLOPs, while MHEALTH and PAMAP2 show moderate savings around 25-35% at higher quantiles. These measurements represent the theoretical speedup from replacing full forward passes with single matrix-vector multiplications, excluding cache lookup overhead and linearization costs. Optimizing these auxiliary operations remains future work, but the results demonstrate substantial potential for computational savings in embedded deployments.

Implications for embedded deployment. These results confirm that ReLU networks processing temporal sensor data repeatedly activate a limited set of linear regions, enabling efficient caching. The hypersphere cache yields up to 55 % theoretical computational reduction with no accuracy loss, underscoring its suitability for resource-constrained IoT devices requiring long-term, energy-efficient inference.

6 Threats to Validity

While our experiments demonstrate the potential energy savings of using caching to reduce FLOPs on microcontrollers without hardware FPUs, there are several threats to the validity of our results:

- **Limited practical validation:** Our FLOP reduction estimations only give a rough estimate for compute and energy savings. More practical experiments are needed, where the neural network models and caching mechanism are deployed on real hardware, and energy consumption is measured for users performing activities similar to those in the datasets. Such measurements would provide more accurate and reliable validation of the energy savings.
- **Feature dimensionality and model scaling:** Our current approach relies on a relatively small number of input features derived from sensor windows. Scaling to higher-dimensional input spaces or more complex features may increase the number of FLOPs, potentially offsetting the energy savings from caching. Additionally, large input dimensions could require larger neural networks or more hidden layers to maintain accuracy, which may further impact energy consumption. The cache size is also dependent on the dimensions; the additional RAM needed for high-dimensional inputs could force one to flush the cache

https://doi.org/10.34749/3061-1008.2025.1

more often, leading to a higher number of cache misses and further amplifying the problem.
- **Cache effectiveness:** The benefits of caching depend on temporal redundancy in the sensor data and the chosen radius quantile, both impacting the cache hit rate. Different activity patterns or datasets may reduce redundancy, diminishing the energy savings.

Addressing these threats in future work will strengthen the validity of our conclusions and provide a more accurate assessment of the trade-offs involved in reducing floating-point operations for embedded neural network inference.

7 Conclusion

This work introduced *Neural Caching*, a region-based inference mechanism that leverages the piecewise-linear structure of ReLU networks to accelerate neural inference on resource-constrained IoT devices. By caching affine mappings corresponding to previously activated ReLU regions, the proposed *Hypersphere Cache* enables subsequent queries to be resolved through a single matrix multiplication instead of a full forward pass. We demonstrated that this approach yields a consistent reduction in necessary floating-point operations, given low input dimensionality, while maintaining classification fidelity across multiple HAR datasets.

Unlike compression or quantization methods, our technique operates entirely at inference time, requiring no retraining or architectural modifications. The empirical results confirm that ReLU-MLPs revisit a limited subset of activation patterns during continuous sensing, enabling efficient reuse without accuracy degradation. Moreover, because caching reduces the number of active operations and memory accesses, it directly contributes to lower energy consumption and thus extends hardware longevity—an essential goal for long-lived IoT deployments.

Future work includes integrating the cache into mixed-precision and hardware-accelerated inference pipelines, exploring adaptive cache replacement strategies under dynamic workloads, and extending the approach to convolutional and transformer architectures. By coupling neural caching with emerging low-power accelerators, sustainable and high-performance AI inference on embedded systems becomes increasingly feasible, advancing the vision of autonomous and energy-aware IoT intelligence.

References

Youva Addad, Alexis Lechervy, and Frédéric Jurie. 2023. Multi-Exit Resource-Efficient Neural Architecture for Image Classification with Optimized Fusion Block. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. IEEE, Paris, France, 1486–1491. https://openaccess.thecvf.com/content/ICCV2023W/

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3–4.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. 2018. Understanding Deep Neural Networks with Rectified Linear Units. arXiv:1611.01491 [cs.LG] https://arxiv.org/abs/1611.01491

Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. 1997. Locally weighted learning. *Artificial intelligence review* 11, 1 (1997), 11–73.

Arjun Balasubramanian, Adarsh Kumar, Yuhan Liu, Han Cao, Shivaram Venkataraman, and Aditya Akella. 2021. Accelerating Deep Learning Inference via Learned Caches. *ArXiv* abs/2101.07344 (2021). https://api.semanticscholar.org/CorpusID:231639254

Oresti Baños, Rafael García, Juan Antonio Holgado Terriza, Miguel Damas, Héctor Pomares, Ignacio Rojas, Alejandro Saez, and Claudia Villalonga. 2014. mHealthDroid: A Novel Framework for Agile Development of Mobile Health Applications. In *International Workshop on Ambient Assisted Living and Home Care*. https://api.semanticscholar.org/CorpusID:9757468

Davis Blalock and John Guttag. 2021. Multiplying matrices without multiplying. In *International Conference on Machine Learning*. PMLR, 992–1004.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. 2017. Adaptive neural networks for efficient inference. In *International conference on machine learning*. PMLR, 527–536.

Boris Hanin and David Rolnick. 2019. Deep ReLU networks have surprisingly few activation patterns. *Advances in Neural Information Processing Systems* 32 (2019).

Johnny Joyce and Jan Verschelde. 2024. Algebraic representations for faster predictions in convolutional neural networks. In *International Workshop on Computer Algebra in Scientific Computing*. Springer, 161–177.

Johnny Joyce and Jan Verschelde. 2025. Computing Linear Regions in Neural Networks with Skip Connections. *arXiv preprint arXiv:2509.15441* (September 2025). https://arxiv.org/abs/2509.15441

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

Daniël Kraak, Mottaqiallah Taouil, Said Hamdioui, Pieter Weckx, Francky Catthoor, Abhijit Chatterjee, Adit Singh, Hans-Joachim Wunderlich, and Naghmeh Karimi. 2018. Device aging: A reliability and security concern. In *2018 IEEE 23rd European Test Symposium (ETS)*. IEEE, 1–10.

Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. 2011. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.* 12, 2 (March 2011), 74–82. doi:10.1145/1964897.1964918

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. 2014. On the number of linear regions of deep neural networks. *Advances in neural information processing systems* 27 (2014).

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. 2016. Exponential expressivity in deep neural networks through transient chaos. In *NeurIPS*.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. 2017. On the Expressive Power of Deep Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 2847–2854. https://proceedings.mlr.press/v70/raghu17a.html

Attila Reiss and Didier Stricker. 2012. Introducing a New Benchmarked Dataset for Activity Monitoring. *2012 16th International Symposium on Wearable Computers* (2012), 108–109. https://api.semanticscholar.org/CorpusID:10337279

Ovishake Sen, Chukwufumnanya Ogbogu, Peyman Dehghanzadeh, Janardhan Rao Doppa, Swarup Bhunia, Partha Pratim Pande, and Baibhab Chatterjee. 2024. Look-Up Table based Neural Network Hardware. arXiv:2406.05282 [cs.AR] https://arxiv.org/abs/2406.05282

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2464–2469.

Matus Telgarsky. 2016. Benefits of depth in neural networks. In *Conference on learning theory*. PMLR, 1517–1539.

Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. 2018. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th annual international conference on mobile computing and networking*. 129–144.

Syed Faizan Alam Zaidi, Junaid Arshad, and Syed Farrukh Alam Zaidi. 2024. A Review on Performance Prediction Models for Battery Life in IoT Networks. *International Journal of Multidisciplinary Sciences and Engineering (IJMSE)* 15, 3 (July 2024), 13–19. https://www.ijmse.org/Volume15/Issue3/paper4_15_3.pdf

  https://doi.org/10.34749/3061-1008.2025.1