

LipCache: A Local Inference Proxy with Certified Caching for Edge Image Classification Service

Zhengzhe Xiang, Yinlin Chen, Fuli Ying, Binbin Zhou, Hailiang Zhao *Member, IEEE*,
Schahram Dustdar *Fellow, IEEE*

Abstract—As edge-side vision services continue to expand toward low-latency, high-throughput scenarios, reducing the inference cost of vision models without sacrificing reliability has become a central concern. Existing semantic caching methods largely rely on empirical similarity thresholds; while such thresholds improve hit rates, they tend to introduce silent misclassifications near decision boundaries. To address this, we propose *LipCache*, a certified semantic caching framework for image classification. Without modifying the existing deployed main model, *MainNet*, the framework introduces a lightweight network, *GuardNet*, that maps inputs into a low-dimensional feature space subject to a Lipschitz constraint. It then computes a per-sample certified reuse radius from the local classification margin and the spectral norm of the classification head. At runtime, a cached result is reused only when the query feature falls inside the certified reuse ball; otherwise, the query falls back to *MainNet*. Thus, cache hits are transformed from empirical threshold tests into geometric certification decisions with explicit theoretical boundaries. Across standard image classification tasks like CIFAR, Tiny-ImageNet, and SVHN, *LipCache* achieves a measured speedup of up to $1.65\times$ with limited end-to-end accuracy degradation, while all accepted cache hits satisfy the *GuardNet*-side certified-consistency condition. Furthermore, an enhanced *GuardNet* training recipe substantially improves cache hit rates in the Tiny-ImageNet multi-class extension while maintaining a certified-consistency rate of 100%. These results demonstrate that per-sample certified reuse can reduce main-model fallback while preserving theoretical consistency, providing a feasible approach to reliable cache-assisted inference at the edge.

Index Terms—Edge intelligence, image classification, semantic caching



1 INTRODUCTION

DEEP-neural-network-based vision services have been widely deployed across diverse domains such as autonomous-driving perception, intelligent manufacturing inspection, real-time surveillance, and mobile augmented reality. The dual demand of these services for inference accuracy and response speed has made high-accuracy neural models a de facto standard component. However, the parameter count and computational complexity of such models keep growing, so that a single inference already consumes considerable compute and memory resources.

This tension is further amplified in industrial edge scenarios. Unlike occasional photo recognition on a phone, industrial edge devices—such as production-line cameras and roadside perception units—typically operate continuously at high sampling rates, injecting a large volume of concurrent queries into the inference service within very short intervals. When these queries converge at the cloud, they readily trigger spikes in queries-per-second, network

congestion, and server overload, severely threatening the overall service availability and tail latency. In other words, the root cause of the problem lies not only in the high cost of a single inference, but more critically in the fact that a large number of semantically redundant queries are repeatedly executed without discrimination.

The rise of the edge computing paradigm offers a structural opportunity to mitigate this risk. By pushing part of the computation or storage capability to edge nodes close to the data source, the system can potentially absorb a fraction of the query load locally, thereby reducing its dependence on the remote cloud link. New challenges arise, however: the latency and memory budgets of edge hardware are extremely tight and far from sufficient to host a full-sized model comparable to its cloud counterpart [1], [2], [3], [4]. Directly deploying a high-accuracy classifier on an edge device almost inevitably faces a sharp conflict between accuracy and resources.

To address this tension, existing work proceeds along roughly two lines. The first line aims to reduce the execution cost of the model at the edge, including compact architecture design, model compression, and early-exit strategies [5], [6], [7]. These methods substantially improve edge deployability, yet their unit of optimization remains “how to run the model more cheaply”; they do not touch upon a more fundamental question: for the many semantically repetitive queries, is it really necessary to run the model every time?

The *cache-assisted inference* starts precisely from this gap. It exploits the pervasive redundancy of visual patterns in edge workloads and reuses previously computed results to skip part of the model invocations directly [8], [9], [10],

This work was partially supported by the National Key R&D Program of China (No. 2025YFG0100700) and the Supercomputing Center of Hangzhou City University. Additional support was provided by Grant CNS2023-144359, funded by MICIU/AEI/10.130391101100011033, and by the European Union NextGeneration EU/PRTR.

- Z. Xiang, Y. Chen, F. Ying, B. Zhou are with Hangzhou City University, Hangzhou, China (e-mail: {xiangzz@hzcu, chenyinlin@stu.hzcu, yingfuli@stu.hzcu, bbzhou@hzcu}.edu.cn).
- H. Zhao is with Zhejiang University, Hangzhou, China (e-mail: hliangzhao@zju.edu.cn).
- S. Dustdar is with ICREA, Barcelona, Spain, and the Distributed Systems Group, TU Wien, Vienna, Austria (e-mail: dustdar@dsg.tuwien.ac.at).
- Copyright (c) 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for any other use, by sending a request to pubs-permissions@ieee.org.

[11], [12]. However, current reuse strategies suffer from a structural weakness in reliability: exact matching is extremely sensitive to compression, cropping, and illumination changes, and can hardly enable effective reuse in real deployments; semantic caching, although it enlarges the reuse region, mostly relies on heuristic similarity thresholds and provides no formal guarantee on the correctness of the reused labels. Consequently, cache hits near decision boundaries may introduce silent misclassifications and uncontrollable accuracy loss. A caching strategy that combines an explainable safety boundary with practical reuse efficiency is still missing.

This paper proposes `LipCache`, a certified semantic caching framework that comprehensively exploits the storage and computation capability of edge devices. Its core design principle is modularity: rather than modifying the deployed high-accuracy main model (`MainNet`), it introduces a lightweight guard network, `GuardNet`, as a new branch to construct the semantic cache. Through operator-norm-controlled convolutions and spectral normalization, `GuardNet` maps inputs to a smooth low-dimensional feature space [13], [14], and, starting from the local classification margin and the Lipschitz bound of the classification head, computes a per-sample *certified reuse radius* offline for each cached sample. Online, only when the query feature falls inside the certified ball of some cached sample, namely when their distance is less than the certified reuse radius, does the system reuse the stored label; otherwise it falls back to `MainNet` to preserve accuracy. Thereby, a cache hit is converted from an empirical threshold decision into a geometric certification decision with an explicit theoretical boundary.

We validate the core certification mechanism of `LipCache` on benchmarks with markedly different statistical structure: CIFAR (natural images), Tiny-ImageNet (higher-resolution fine-grained inter-class differences), and SVHN (domain-shifted digits and characters). This choice is designed to isolate the stability of the certified radius under different task geometries, rather than to exhaust all visual task scales; larger-scale settings (e.g., full ImageNet) and non-i.i.d. streaming scenarios are left as open directions discussed in Section 7. On these three tasks, the average hit rates under the tight certified radius reach 0.342, 0.472, and 0.502, respectively, with end-to-end accuracies of 0.921, 0.867, and 0.960, and corresponding measured speedups of 1.32 $\times$, 1.31 $\times$, and 1.65 $\times$; moreover, the certified-consistency rate of all results is 100%. Comparisons with empirical thresholds and the global nearest-neighbor baseline show that the certified radius is the only reuse boundary that can stably preserve certified consistency across the three tasks. Furthermore, in the Tiny-ImageNet multi-class extension, an enhanced `GuardNet` training recipe improves the hit rates of the 20/30/50-class tasks from 0.056/0.005/0.0004 to 0.423/0.254/0.124, while both the end-to-end accuracy and the certified consistency are preserved. The main contributions are as follows:

- 1) We propose a dual-model cache-assisted inference framework that comprehensively exploits the storage and computation capability of edge devices and, without modifying `MainNet`, makes certified reuse decisions through a lightweight `GuardNet`.

- 2) Starting from the local classification margin and the Lipschitz bound of the `GuardNet` classification head, we derive a per-sample certified reuse radius in the feature space, providing a computable and interpretable rule for safe semantic reuse.
- 3) We systematically evaluate `LipCache` across multiple benchmark tasks, multi-class extensions, and edge-deployment boundaries, verifying the effectiveness of the certified radius as the only cross-task stable certified boundary and revealing the potential of the enhanced training recipe in enlarging the certified reuse region.

2 RELATED WORK

Edge image classification faces a structural tension: high-accuracy neural models keep growing in their computational and memory demands, whereas edge devices must satisfy tight latency, energy, storage, and connectivity constraints. A practical edge system therefore requires more than just a smaller classifier. It must reduce computation, exploit redundancy among correlated inputs, and decide when a low-cost shortcut is safe rather than merely fast. The literature addresses these needs along three main lines and one important bridge. Efficient edge inference reduces the cost of running a model through compact architectures, adaptive execution, hardware–software co-design, or edge–cloud partitioning. Cache-assisted inference and semantic reuse avoid part of the execution by reusing results from temporally or semantically correlated inputs. Certified nearest-neighbor retrieval forms a bridge, studying when a nearest-neighbor decision in an embedding space remains invariant under perturbation. Lipschitz control and certification provide the tools that turn margins and continuity bounds into local stability guarantees. `LipCache` connects these lines: it uses a lightweight `GuardNet` as a cache-oriented front end to the high-accuracy `MainNet`, and accepts reuse only inside a per-sample certified reuse radius in the `GuardNet` feature space.

2.1 Inference for Edge Vision Systems

The dominant strategy for edge inference is to reduce the cost per invocation, or to place computation where it can be executed more efficiently. Survey work points out the underlying pressure: the scale and computational complexity of DNNs grow faster than the performance and efficiency gains that embedded platforms can provide [1]. Recent surveys organize the response around efficient architectures, model compression, hardware acceleration, algorithm–hardware co-design, and deployment techniques for on-device or edge–cloud execution [2], [3], [4], [15]. Edge–cloud collaboration further partitions a DNN so that shallow feature extraction runs on the device while deeper layers run on the cloud or an edge server, additionally attending to privacy, communication overhead, and resource allocation [16], [17].

Representative architecture-level methods reduce the cost of the backbone itself. MobileNetV2 uses inverted residual blocks, thin linear bottlenecks, and depthwise separable convolutions to improve mobile classification, detection,

and segmentation across model sizes [5]. MobileNetV3 combines hardware-aware neural architecture search with NeAdapt and manual architectural refinements, yielding both large and small mobile models tuned for phone CPUs [18]. ShuffleNet targets extremely low computational budgets by combining pointwise grouped convolutions with channel shuffling, lowering cost while preserving mobile accuracy [6]. Representative execution-level methods adaptively reduce the average cost: BranchyNet adds side-branch classifiers so that high-confidence samples can exit early, while harder samples continue through deeper layers [7]; MSDNet treats test-time computation as an anytime or budgeted classification problem and reuses multi-scale dense features across multiple early exits [19]. At the collaborative-system level, MAE reduces intermediate-transmission and scheduling costs by activating sparse channel-level experts during DNN inference on edge devices [17], and more recent edge-cloud co-inference combines spike-driven compression with dynamic early-exit to cut both transmission volume and end-to-end latency [20].

These works substantially improve deployability, yet their unit of optimization remains executing the current query—through the model, an early-exit path, or a device-cloud partition. They do not explicitly exploit a per-sample certified reuse region around cached samples to decide whether the current query can entirely skip the main classifier. LipCache is therefore complementary: it keeps MainNet unchanged as a reliable fallback predictor and adds an independent GuardNet-guided cache decision layer to avoid selected MainNet invocations.

2.2 Cache-Assisted Neural Network Inference

Cache-assisted inference follows a different strategy: rather than reducing the cost per invocation, it tries to avoid invocations by reusing outputs or intermediate states of correlated inputs. Exact-match memoization is too brittle for vision tasks, since repeated content may differ due to camera motion, illumination, scale, cropping, or semantic changes. Practical systems therefore extend reuse beyond equality by exploiting temporal locality, video-frame coherence, low-cost filters, intermediate feature maps, or semantic memory.

Shadow Puppets proposes a semantic caching service that breaks the binary choice between pure cloud inference and full edge execution, leveraging edge-side caching to approximate cloud-level accuracy at lower latency and cost [21]. DeepCache targets continuous mobile vision: it uses heuristics inspired by video compression to detect reusable regions in the video input, propagates these reusable regions through the layers of the CNN, works with unmodified models, and reports reductions in inference time and energy [10]. NoScope specializes video analytics to a specific video and object, searching over a cascade of cheap specialized models and difference detectors to mimic a reference network at a chosen accuracy level [9]. DeltaCNN likewise exploits video coherence but at the implementation level: it propagates sparse inter-frame differences through typical CNN layers and accelerates per-frame inference without retraining [12]. Semantic Memory turns reuse toward semantic features, encoding high-dimensional feature maps

into low-dimensional semantic vectors, and uses hierarchical memory and adaptive cache management to accelerate mobile CNN inference with acceptable accuracy loss [11]. More recent work continues to expand cache-assisted inference along several axes: synergistic lazy loading and layer-wise caching reduce the cold-start and loading cost of serverless edge models [22]; task-aware expert caching decomposes multitask models into modular experts for fine-grained reuse [23]; and distributed inference caching has been formalized as a submodular maximization problem under storage constraints [24]. The closest line on certified retrieval is RetrievalGuard, which studies provably robust 1-nearest-neighbor image retrieval. Given a base retrieval model and a query, RetrievalGuard constructs a smoothed retrieval model, analyzes the 1-NN search process in the high-dimensional embedding space, and derives a computable ℓ_2 radius within which the Recall@1 result remains unchanged [25].

These systems demonstrate that reuse can be a powerful source of acceleration, yet their acceptance criteria are mainly empirical or workload-specific. Video-based methods rely on temporal continuity or sparse inter-frame differences; specialized cascades rely on a fixed camera/query distribution; semantic memory relies on learned or measured feature similarity and exit policies; certified retrieval certifies the stability of the nearest-neighbor result itself, rather than the safe reuse of a cached payload in an edge inference pipeline. LipCache retains the idea of reuse but changes the acceptance criterion: the cache is built in a dedicated GuardNet feature space, and a reuse is accepted only when the query falls inside the per-sample certified reuse radius of some cached sample, a radius derived from the GuardNet-side margin and Lipschitz bounds.

2.3 Lipschitz Control and Certification

Lipschitz control and certification address the safety problem of efficient inference. Their common strategy is to bound how fast a neural mapping can change, and then use this bound, together with margins or probabilistic arguments, to derive a region within which the prediction remains stable. This body of literature therefore provides the mathematical tools for turning similarity into local consistency statements.

At the layer and architecture level, spectral normalization constrains the operator norm of the weights through a lightweight normalization procedure, originally introduced to stabilize GAN discriminators [14]. For convolutional layers, Sedghi et al. characterize the singular values of standard multi-channel convolutions, enabling efficient computation and projection onto operator-norm balls [13]. Araujo et al. provide a unified algebraic view of 1-Lipschitz neural networks, showing that several layers based on orthogonality, spectra, and convex potentials can be understood through a common semidefinite-programming condition [26]. These works explain how to build or constrain networks whose continuity can be controlled.

At the certification level, CROWN bounds general activation functions with linear or quadratic surrogates to produce certified lower bounds on adversarial distortion [27]; randomized smoothing, in turn, turns a classifier

that remains accurate under Gaussian noise into an L_2 -certified smoothed classifier, extending certification to full-resolution ImageNet [28]. More recent work tightens or extends Lipschitz-based certification. Huang et al. exploit activation states to remove inactive rows and columns from the induced-norm computation, thereby computing efficient local Lipschitz upper bounds [29]. Fazlyab et al. use directional Lipschitz bounds and improved regularization to maximize a differentiable lower bound on the distance to the decision boundary [30]. Hu et al. show that achieving certifiable robustness under a Lipschitz constraint requires careful capacity and data design, including larger Lipschitz-controlled residual dense layers and data augmentation [31]. Wang et al. improve scalability by reformulating semidefinite-programming Lipschitz estimation as an eigenvalue optimization problem [32]; and ECLipsE decomposes the large verification problem into layer-scale subproblems or closed-form relaxations to obtain faster compositional estimates [33].

From the perspective of cache-assisted edge inference, the common limitation of these methods is that they certify model robustness or stability, rather than cache reuse. Their certificates typically answer whether a classifier’s prediction remains invariant under perturbation; they do not decide whether a cached payload should replace an expensive MainNet invocation. LipCache repurposes the same mathematical ingredients for a system-level reuse decision: the GuardNet margin and the GuardNet-side Lipschitz control define a per-sample certified reuse radius in the GuardNet feature space. The resulting guarantee is local to GuardNet, whereas consistency with MainNet is treated as an empirical system property.

In summary, existing work leaves three gaps with respect to reliable edge caching. Edge-inference methods reduce the cost of running a model but do not address whether the model can be skipped. Semantic caching and video reuse exploit redundancy but typically rely on empirical thresholds or workload locality. Certified-retrieval and robustness methods provide stability guarantees, but they do not formulate the cache-hit acceptance criterion as a GuardNet-side reuse certificate coupled with MainNet fallback. LipCache fills exactly this intersection by combining a dedicated GuardNet feature space, a per-sample certified reuse radius for each cached sample, and an empirical evaluation of the consistency between the accepted cache hits and MainNet.

3 PROBLEM DESCRIPTION

3.1 System Setting and Cache Motivation

This section provides a formal model of cache-assisted inference in an edge image classification system, establishing a unified notational foundation for the framework design in Section 4 and the certification analysis in Section 5.

Consider an image classification system whose input space is $\mathcal{X} \subseteq \mathbb{R}^n$ and whose label set is $\mathcal{Y} = \{1, \dots, C\}$. The system may invoke a high-accuracy classifier $M : \mathcal{X} \rightarrow \mathcal{Y}$, but a single inference of M is costly—in edge scenarios this may amount to a combination of GPU time, cloud communication latency, and other overhead. For any input

$x \in \mathcal{X}$, the system ultimately needs to output a prediction $\hat{y} \in \mathcal{Y}$.

A key observation is that edge vision workloads are not composed of mutually independent random samples. The input-label pairs (x, y) are drawn from a joint distribution $\mathcal{D}$ that often exhibits pronounced structural redundancy: influenced by factors such as a fixed camera viewpoint, periodic sampling, and slow scene changes, consecutively arriving queries are highly clustered in pixel space or semantic space, and adjacent inputs frequently share the same label. Under these conditions, executing M in full for every query means that a large amount of computation is wasted on samples whose answers could have been inferred from existing results. This observation points to a natural optimization path: cache a subset of historical predictions, and when a new query is sufficiently similar to some cached sample, directly reuse its label, thereby skipping the invocation of M .

However, the above path faces a fundamental tension. If the reuse condition is too loose, queries near a decision boundary may be incorrectly matched to an inappropriate cached label, leading to silent accuracy loss; if the reuse condition is too strict, cache hits will be so rare that the savings in M invocations become negligible. The core challenge is therefore not “whether to introduce a cache,” but how to define a reuse rule that simultaneously achieves a sufficient hit rate to yield meaningful system gains and constrains the resulting prediction errors to a controllable range.

3.2 Formalization of the Optimization Objective

To precisely characterize the above trade-off, we abstract cache-assisted inference as an optimization problem over deployment policies.

A deployment policy π produces two outputs for each input $x \in \mathcal{X}$: a predicted label $\hat{y}(x) \in \mathcal{Y}$, and a binary routing decision $\rho(x) \in \{\text{cache}, \text{invoke}\}$, the latter indicating whether the prediction comes from cache reuse or from a direct inference of M .

Let $(x, y) \sim \mathcal{D}$ denote a test sample drawn from the data distribution. The system accuracy of policy π is defined as

$$\text{Acc}_{\text{sys}}(\pi) \triangleq \Pr_{(x,y) \sim \mathcal{D}}[\hat{y}(x) = y], \quad (1)$$

and the standalone accuracy of M is $\text{Acc}_M \triangleq \Pr_{(x,y) \sim \mathcal{D}}[M(x) = y]$. In practice, the above distributional expectations are approximated by empirical frequencies on a reference validation or test set.

The invocation rate of policy π on M is defined as

$$\text{InvokeRate}_M(\pi) \triangleq \Pr_{(x,y) \sim \mathcal{D}}[\rho(x) = \text{invoke}]. \quad (2)$$

This quantity directly determines the degree of cost savings on M inference—the lower the invocation rate, the higher the fraction of M invocations that are skipped.

Cache-assisted inference can thus be formalized as the following constrained optimization problem:

$$\begin{aligned} \min_{\pi} \quad & \text{InvokeRate}_M(\pi) \\ \text{s.t.} \quad & \text{Acc}_{\text{sys}}(\pi) \geq \text{Acc}_M - \Delta_{\text{acc}}, \end{aligned} \quad (3)$$

where $\Delta_{\text{acc}} \geq 0$ is a user-specified accuracy tolerance. A smaller Δ_{acc} enforces a more conservative reuse strategy,

tending to depress the hit rate to reduce accuracy risk; a larger Δ_{acc} allows more aggressive cache reuse, trading potential accuracy for a lower M invocation frequency.

4 THE LIPCACHE

Section 3 formalizes cache-assisted inference as a deployment-level optimization problem that minimizes the main-model invocation rate under an accuracy-tolerance constraint. This section gives a concrete instantiation of that problem—the `LipCache` framework. Its workflow proceeds in three stages: first, a Lipschitz-constrained lightweight guard network, `GuardNet`, is trained offline (Section 4.2); next, a feature cache is built on the trained `GuardNet`, and each cached sample is assigned a per-sample certified reuse radius (Section 4.3); finally, in the online phase, `GuardNet` guides cache-hit detection and the fallback decision (Section 4.4). We first establish the unified notation.

4.1 Setup and Notation

`LipCache` adopts a dual-model design, comprising a high-accuracy main model M and a lightweight guard model G . The main model provides the reference prediction:

$$\hat{y}_M(x) = \arg \max_{c \in \mathcal{Y}} M_c(x). \quad (4)$$

The guard model maps each input to a low-dimensional feature space, $G: \mathcal{X} \rightarrow \mathbb{R}^d$. Let $z = G(x)$; an affine classifier operates in this space:

$$s(z) = Wz + b, \quad (5)$$

where $W \in \mathbb{R}^{C \times d}$, $b \in \mathbb{R}^C$, yielding the `GuardNet` prediction

$$\hat{y}_G(x) = \arg \max_{c \in \mathcal{Y}} s_c(G(x)). \quad (6)$$

We denote the full logit map by $F_G(x) = s(G(x))$ and the affine head by $f(z) = Wz + b$. When two logit vectors are subtracted, the bias term cancels, so it does not affect the Lipschitz constant.

For each cached sample, the certified reuse radius is derived from the local classification margin and the Lipschitz bound of f (Section 5). At runtime, a query that falls inside the certified reuse region of some cached sample reuses the stored *payload* (the label returned upon a cache hit); otherwise the system falls back to M . Table 1 summarizes the notation used.

Each cache entry is a triple $(z_i, \tilde{y}_i, r_i)$, containing a `GuardNet` feature, a payload label, and a certified reuse radius. Figure 1 shows the offline training and feature-constraint pipeline, and Figure 2 shows the online inference.

4.2 Offline GuardNet Training

With the notation established, this subsection answers the first core question: how to train a `GuardNet` so that its learned feature geometry both supports discriminative classification and provides a sufficient certified reuse region for subsequent certified caching. To this end, the training objective is designed as a layered framework:

$$L_{\text{total}} = (1 - \alpha)L_{\text{CE}} + \alpha L_{\text{distill}} + \lambda_m L_{\text{margin}}, \quad (7)$$

TABLE 1
Main symbols used in the formulation.

Symbol	Meaning
$\mathcal{X}, \mathcal{Y}$	Input space and label set
M	High-accuracy main model (<code>MainNet</code>)
G	Feature map of the guard model (<code>GuardNet</code>)
W	Weight matrix of the <code>GuardNet</code> affine classifier
d	Dimensionality of the <code>GuardNet</code> feature space
L_G	Lipschitz constant of the <code>GuardNet</code> feature map G
$\sigma_{\max}(W)$	Spectral norm of W
z_i	<code>GuardNet</code> feature of cached sample x_i
$\tilde{y}_i^G$	<code>GuardNet</code> prediction of x_i
$\tilde{y}_i$	Cached payload associated with x_i
r_i	Certified reuse radius of cached feature z_i
z_{new}	<code>GuardNet</code> feature of a new input
m_i	<code>GuardNet</code> classification margin of x_i
$\mathcal{C}$	Feature cache maintained by the system

where $\alpha \in [0, 1]$ balances direct supervision against an optional distillation signal from M , and λ_m controls the margin-regularization strength. When $\alpha = 0$, the objective reduces to the pure cross-entropy training used in the 10-class main experiments of this paper; when $\alpha > 0$, it can be used for additional distillation ablations or teacher-alignment experiments. The above objective subsumes three complementary training signals, which we elaborate on one by one below, starting with the structural constraint that supports them.

Lipschitz control. The theoretical certificate in Section 5 requires the feature map to satisfy $\text{Lip}(G) \leq 1$. To this end, `GuardNet` imposes an operator-norm constraint on all parameterized layers. For a convolutional layer whose kernel is $K \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k_1 \times k_2}$, the Lipschitz constant of the convolution operator is given by its frequency-domain representation:

$$\text{Lip}(K) = \max_{\omega} \sigma_{\max}(\hat{K}(\omega)), \quad (8)$$

where $\hat{K}(\omega)$ is the frequency-domain matrix obtained by zero-padding the kernel and applying a two-dimensional real FFT. Dividing the kernel by $\max(\text{Lip}(K), 1)$ before each forward pass enforces the 1-Lipschitz constraint [13]. Linear layers use standard spectral normalization [14]; ReLU, non-overlapping average pooling, and the final tanh projection are all non-expansive [26]. Since the Lipschitz constant of each individual layer does not exceed 1, by submultiplicativity of operator norms under composition, the overall guard network satisfies $\text{Lip}(G) \leq 1$.

Distillation loss (optional). To optionally leverage the knowledge of M , one may minimize the KL divergence between the softened outputs of M and F_G at temperature T :

$$L_{\text{distill}} = T^2 \text{KL}(\text{Softmax}(\frac{M(x)}{T}) \parallel \text{Softmax}(\frac{F_G(x)}{T})). \quad (9)$$

The standard cross-entropy L_{CE} with the ground-truth label y preserves discriminative performance; the main experiments in this paper use $\alpha = 0$, i.e., this term is disabled.

Margin regularization. The two terms above guarantee the discriminative accuracy of `GuardNet`, whereas margin regularization directly serves cache efficiency: the certified reuse radius r_i is proportional to the `GuardNet` classification margin (see Section 5), so explicitly encouraging large

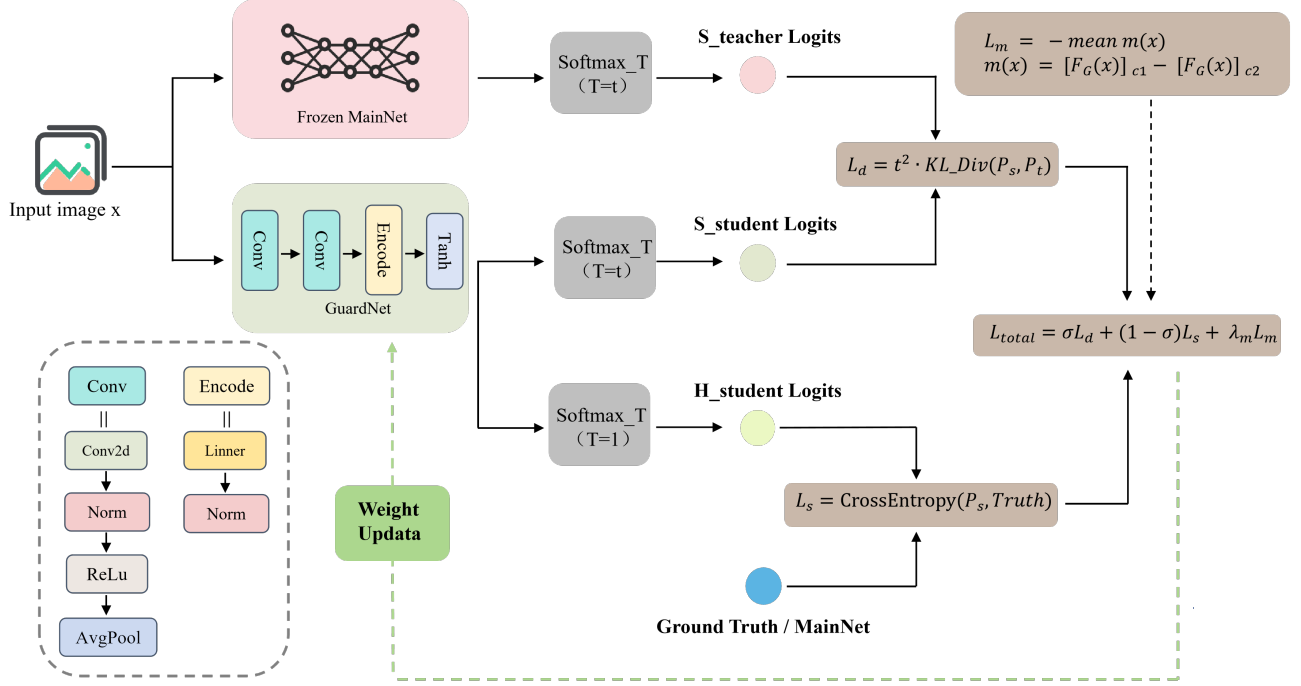


Fig. 1. Offline training pipeline of GuardNet. Lipschitz control on the convolutions and spectral normalization on the linear layers are used to constrain the GuardNet feature map, thereby supporting the subsequent GuardNet-side certification; the main model may provide reference predictions when needed, but it is not a required training signal in the three 10-class main experiments of this paper.

margins is a key lever for improving the hit rate. Let c_1 and c_2 denote the indices of the top-1 and top-2 logits:

$$m(x) = [F_G(x)]_{c_1} - [F_G(x)]_{c_2}. \quad (10)$$

To avoid blindly enlarging the margin on already misclassified samples, the regularizer is applied only to the correctly classified subset of a batch $\mathcal{B}$:

$$\mathcal{S}_{\text{corr}} = \left\{ (x, y) \in \mathcal{B} \mid \arg \max_{c \in \mathcal{Y}} [F_G(x)]_c = y \right\}, \quad (11)$$

$$L_{\text{margin}} = -\frac{1}{\max(1, |\mathcal{S}_{\text{corr}}|)} \sum_{(x, y) \in \mathcal{S}_{\text{corr}}} m(x). \quad (12)$$

This selectively enlarges the margins around samples that are already correctly predicted, and these margins directly determine the certified cache radius.

Enhanced training recipe (optional). For multi-class extensions at higher class counts, cross-entropy alone or a single margin term is often insufficient to sustain a usable certified radius. We therefore additionally consider an enhanced training recipe: beyond the core objective above, it further adds a center constraint that promotes intra-class compactness and an additional inter-class separation term that promotes inter-class separation, while optionally retaining the distillation signal. These additional terms only change the feature geometry learned by GuardNet; they do not change the online hit rule of the cache, nor do they change the certification criterion in Section 5.

In the main-text experiments, the three 10-class main results use cross-entropy supervision only ($\alpha = 0, \lambda_m = 0$); the CIFAR-10 training-objective ablation examines the independent effects of the margin and center constraints, respectively; and the Tiny-ImageNet multi-class extension uses

the full enhanced recipe comprising the margin, interclass, center, and distill terms.

4.3 Feature Cache Construction

Once GuardNet is trained, the framework enters the offline cache-construction phase. Its task is to select a set of high-quality cache entries from the reference set $\mathcal{D}_{\text{ref}}$, so that it can cover the queries in the test distribution as much as possible under a given capacity budget.

For each candidate sample x_i , we first compute $z_i = G(x_i)$, the GuardNet label $y_i^G = \arg \max_c [F_G(x_i)]_c$, and the payload $\tilde{y}_i$ selected by the construction strategy. This paper considers three payload semantics. The first is the *proxy label*, which directly stores $\tilde{y}_i = y_i^G$; in this case the cached payload is naturally aligned with the certificate, and it is the default setting adopted in the main-text experiments. The second is the *MainNet* prediction scheme, which stores $\tilde{y}_i = \hat{y}_M(x_i)$; the third is the *ground-truth* scheme, which stores $\tilde{y}_i = y_i$. The latter two are mainly used to analyze the effect of varying the payload source on system behavior.

The GuardNet margin $m_i = [F_G(x_i)]_{c_1} - [F_G(x_i)]_{c_2}$ is used to compute the certified radius r_i . Depending on the radius mode ρ , this paper supports two choices: the conservative radius $r_i = m_i / (\sigma_{\max}(W) \sqrt{2})$ (Lemma 5.1) or the tighter per-pair radius r_i^* (Remark 5.1). Candidates with non-positive radius are discarded outright.

The certificate is attached to y_i^G . When the cache-center label consistency $y_i^G = \tilde{y}_i$ holds, the returned payload is directly aligned with the certificate; otherwise the reliability of the payload is assessed empirically. To maximize the coverage under a fixed budget, the selection strategy balances radius size against feature-space diversity. Algorithm 1 summarizes this construction process.

Algorithm 1 Offline construction of the feature store.

Require: Reference set $\mathcal{D}_{\text{ref}}$, models M, G, W , cache budget B_C

Require: Payload construction rule, payload source (proxy, MainNet M prediction, or label y), cache selection strategy $\mathcal{S}$

Require: Radius mode ρ , optional cache-center label-consistency flag

Ensure: Feature cache $\mathcal{C}$

- 1: Initialize candidate pool $\mathcal{P} \leftarrow \emptyset$
- 2: Compute $\sigma_{\max}(W)$
- 3: **if** $\sigma_{\max}(W) \leq 0$ **then**
- 4: Report that the GuardNet classification head is degenerate and stop construction
- 5: **end if**
- 6: **for** each candidate sample $x_i \in \mathcal{D}_{\text{ref}}$ **do**
- 7: Compute $z_i = G(x_i)$ and GuardNet logits $F_G(x_i)$
- 8: Compute GuardNet label y_i^G , payload $\tilde{y}_i$, and margin m_i
- 9: Compute certified radius r_i using radius mode ρ
- 10: **if** cache-center label consistency is required and $y_i^G \neq \tilde{y}_i$ **then**
- 11: Skip x_i
- 12: **else if** $r_i \leq 0$ **then**
- 13: Skip x_i
- 14: **else**
- 15: Add candidate entry $(z_i, \tilde{y}_i, r_i)$ to $\mathcal{P}$
- 16: **end if**
- 17: **end for**
- 18: Use $\mathcal{S}$ to select at most B_C entries from $\mathcal{P}$
- 19: The selection process balances certified radius and feature diversity
- 20: Store the selected entries in $\mathcal{C}$
- 21: **return** $\mathcal{C}$

4.4 Online Inference and Cache-Hit Detection

After offline training and cache construction are complete, LipCache enters the online inference state. At runtime (Figure 2), each input x_{new} is encoded as $z_{\text{new}} = G(x_{\text{new}})$, and the hit set is

$$\mathcal{H}(z_{\text{new}}) = \{i \mid \|z_{\text{new}} - z_i\|_2 < r_i\}. \quad (13)$$

If $\mathcal{H} \neq \emptyset$, it takes the payload from the nearest hit entry:

$$i^*(x_{\text{new}}) = \arg \min_{i \in \mathcal{H}(z_{\text{new}})} \|z_{\text{new}} - z_i\|_2. \quad (14)$$

The system predicts

$$\hat{y}_{\text{sys}}(x_{\text{new}}) = \begin{cases} \tilde{y}_{i^*(x_{\text{new}})}, & \mathcal{H}(z_{\text{new}}) \neq \emptyset, \\ \hat{y}_M(x_{\text{new}}), & \mathcal{H}(z_{\text{new}}) = \emptyset. \end{cases} \quad (15)$$

On a miss, the system falls back to M ; if write-back is enabled, a missed sample may be added to the cache under the same construction strategy.

The computational overhead of the above pipeline directly determines the practical gain of the framework. Let C_G and C_M denote the per-query cost of GuardNet and MainNet, respectively. When maintaining N cache entries

Algorithm 2 Online inference with cache reuse.

Require: Query batch $\{x_k\}_{k=1}^B$, main model M , trained GuardNet encoder G and classification head W , feature cache $\mathcal{C} = \{(z_i, \tilde{y}_i, r_i)\}_{i=1}^N$, optional write-back flag

Ensure: Predictions $\{\hat{y}_{\text{sys}}(x_k)\}_{k=1}^B$

- 1: Compute GuardNet features $z_k \leftarrow G(x_k)$ for all k
- 2: Compute pairwise distances $D_{k,i} = \|z_k - z_i\|_2$
- 3: **for** $k = 1$ to B **do**
- 4: Construct hit set $\mathcal{H}(z_k) = \{i \mid D_{k,i} < r_i\}$
- 5: **if** $\mathcal{H}(z_k) \neq \emptyset$ **then**
- 6: $i_k^* = \arg \min_{i \in \mathcal{H}(z_k)} D_{k,i}$
- 7: $\hat{y}_{\text{sys}}(x_k) \leftarrow \tilde{y}_{i_k^*}$
- 8: **else**
- 9: $\hat{y}_{\text{sys}}(x_k) \leftarrow \arg \max M(x_k)$
- 10: **if** write-back is enabled and the required payload source is available **then**
- 11: Compute radius and payload for x_k using GuardNet logits and W ; update $\mathcal{C}$
- 12: **end if**
- 13: **end if**
- 14: **end for**
- 15: **return** $\{\hat{y}_{\text{sys}}(x_k)\}_{k=1}^B$

in a d -dimensional feature space, the expected cost per query is

$$\mathbb{E}[\text{Cost}] = O(C_G + Nd + \text{InvokeRate}_M C_M). \quad (16)$$

When $C_G + Nd$ is much smaller than C_M , the system can reduce the overall overhead in expectation even if the hit rate is not high. For large caches, hierarchical indexing or a vector database can further reduce the search complexity to sublinear, without changing the decision rule itself. Algorithm 2 summarizes the online pipeline.

Combining the three stages above, LipCache instantiates the abstract deployment policy of Section 3 as a parametric policy family $\pi = (\theta_G, \mathcal{C}, \rho, \mathcal{R}, \omega)$, where θ_G is the GuardNet parameter set, $\mathcal{C}$ is the feature cache, ρ is the radius mode, $\mathcal{R}$ is the hit-resolution rule, and ω controls the payload-source and cache-center label-consistency options. Different parameter combinations correspond to different operating points—these are precisely the core objects of the experimental evaluation in Section 6.

5 THEORETICAL ANALYSIS

Section 4 gives LipCache a complete engineering pipeline, yet its core decision—the cache-hit test—is still an empirical definition: $\|z_{\text{new}} - z_i\|_2 < r_i$. This section injects theoretical content into that decision: starting from the Lipschitz property of the classifier in the GuardNet feature space and the local classification margin, we derive a per-sample certified reuse radius r_i and prove that any query falling inside this radius necessarily preserves the same GuardNet prediction label as the cache center. Thereby, the accuracy constraint Δ_{acc} in the optimization of Section 3 obtains a computable lower bound at the GuardNet level—as long as cache hits occur strictly within the certified radius, label consistency on the GuardNet side is guaranteed mathematically.

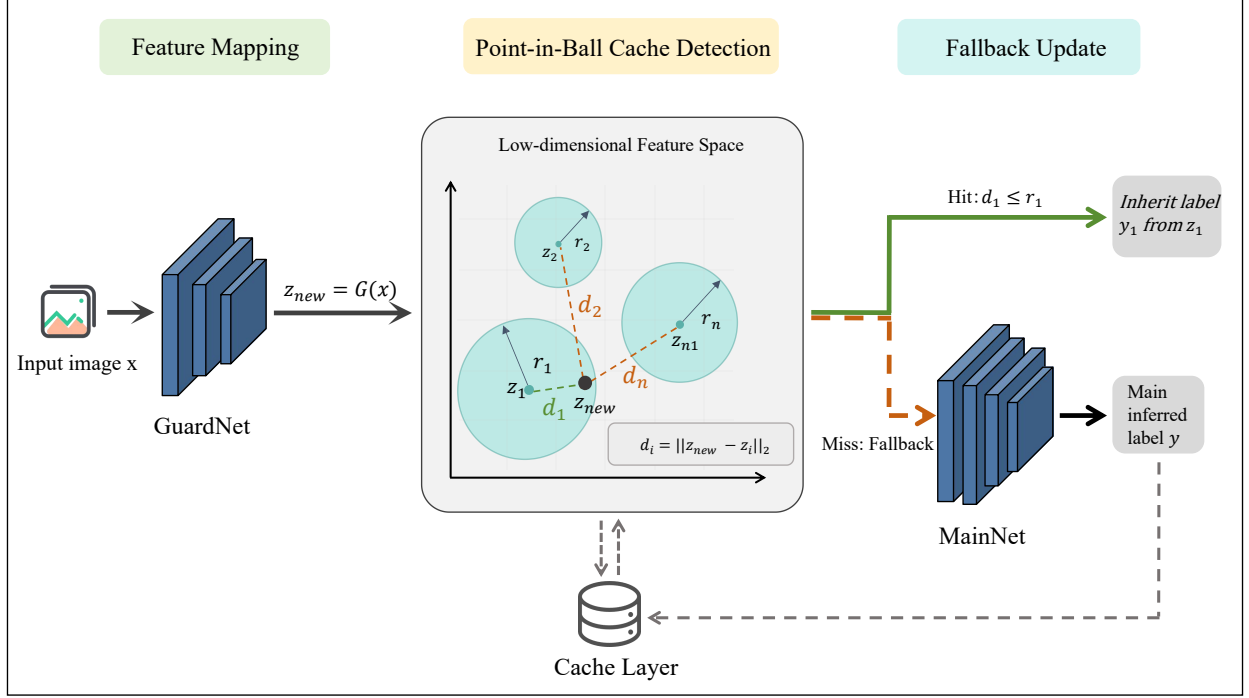


Fig. 2. Online inference pipeline of LipCache. For a new input, GuardNet extracts a compact feature and uses it to search the cache for certified reuse. A cache hit returns the stored payload; a miss triggers a MainNet fallback.

5.1 Lipschitz Setup

The structure of the certified reuse radius can be intuitively understood as “classification margin divided by the Lipschitz bound”: the larger the margin, the larger the admissible perturbation; the smoother the mapping, the smaller the logit change induced by the same input variation. We therefore first establish the Lipschitz bound of the GuardNet composite map $F_G = f \circ G$.

All Lipschitz constants are measured under the ℓ_2 norm. The feature encoder G has a finite Lipschitz constant

$$L_G \triangleq \text{Lip}(G), \quad (17)$$

and the per-layer spectral-normalization construction of Section 4.2 enforces $L_G \leq 1$. For the affine classification head $f(z) = Wz + b$, noting that the bias term cancels automatically in a logit difference, we have

$$\|f(z_1) - f(z_2)\|_2 = \|W(z_1 - z_2)\|_2 \leq \sigma_{\max}(W)\|z_1 - z_2\|_2, \quad (18)$$

so f is $\sigma_{\max}(W)$ -Lipschitz. By submultiplicativity,

$$\text{Lip}(F_G) = \text{Lip}(f \circ G) \leq L_G \sigma_{\max}(W). \quad (19)$$

Under the constraint $L_G \leq 1$, this simplifies to $\text{Lip}(F_G) \leq \sigma_{\max}(W)$. This means that along any direction in the feature space, the rate of change of F_G ’s output is controlled by $\sigma_{\max}(W)$. All certificates below are stated in the GuardNet feature space and depend only on $\sigma_{\max}(W)$ —this is also the reason why the cache-construction phase in Section 4 needs to compute only this single spectral norm.

5.2 Certified Local Label Consistency

With the global Lipschitz bound in hand, we now turn to the local analysis: for a fixed cached sample, within

what neighborhood does its GuardNet prediction remain unchanged.

For a cached sample x_i , let its feature be $z_i = G(x_i)$ and its GuardNet prediction label be $y_i^G = \arg \max_c f_c(z_i)$. Define the local classification margin of this sample as

$$m_i = [f(z_i)]_{y_i^G} - \max_{j \neq y_i^G} [f(z_i)]_j. \quad (20)$$

Geometrically, m_i measures the logit-space margin between z_i and the nearest decision boundary—the larger the margin, the farther the sample lies from a decision boundary, and the larger the perturbation required to flip its label. This margin-based certification idea is widely adopted in the certified-robustness literature [29], [30], [31], [32], [33]. Accordingly, the certified reuse radius is defined as

$$r_i = \frac{m_i}{\sigma_{\max}(W)\sqrt{2}}, \quad (21)$$

where $\sigma_{\max}(W) > 0$; the degenerate case $\sigma_{\max}(W) = 0$ means that the classification head degenerates to a constant map, in which case the reuse radius of any positive-margin sample is unbounded, but this does not occur in practice. The $\sqrt{2}$ factor in the denominator comes from the worst-case analysis that simultaneously bounds all competing classes, and it will be tightened in Remark 5.1.

The following lemma is the core theoretical result of this framework; it establishes the guarantee that “the label is invariant inside the certified ball.”

Lemma 5.1 (Local label consistency). Let z_i be a cached feature with GuardNet label y_i^G and positive margin $m_i > 0$, and assume $\sigma_{\max}(W) > 0$. For any new sample x_{new} whose GuardNet feature is $z_{\text{new}} = G(x_{\text{new}})$, if

$$\|z_{\text{new}} - z_i\|_2 < r_i = \frac{m_i}{\sigma_{\max}(W)\sqrt{2}}, \quad (22)$$

then the GuardNet prediction at z_{new} is still y_i^G .

The core idea of the proof is to use the Lipschitz bound to translate the displacement in feature space into an upper bound on the perturbation in logit space, and then to show that this perturbation is insufficient to cover the original margin m_i .

Proof: Let $\delta = f(z_{\text{new}}) - f(z_i)$. By the Lipschitz property of f ,

$$\|\delta\|_2 \leq \sigma_{\max}(W)\|z_{\text{new}} - z_i\|_2. \quad (23)$$

To preserve the label y_i^G , it suffices to show that for every competing class $j \neq y_i^G$ we have $[f(z_{\text{new}})]_{y_i^G} > [f(z_{\text{new}})]_j$. Substituting $f(z_{\text{new}}) = f(z_i) + \delta$ into the logit difference:

$$\begin{aligned} [f(z_{\text{new}})]_{y_i^G} - [f(z_{\text{new}})]_j &= [f(z_i)]_{y_i^G} + \delta_{y_i^G} - [f(z_i)]_j - \delta_j \\ &= ([f(z_i)]_{y_i^G} - [f(z_i)]_j) - (\delta_j - \delta_{y_i^G}). \end{aligned} \quad (24)$$

By the definition of the margin, $[f(z_i)]_{y_i^G} - [f(z_i)]_j \geq m_i$, so a sufficient condition for preserving the label is

$$\delta_j - \delta_{y_i^G} < m_i. \quad (25)$$

To bound the left-hand side, introduce the auxiliary vector e_{j,y_i^G} : it takes the value 1 at position j , -1 at position y_i^G , and 0 elsewhere. Then $\|e_{j,y_i^G}\|_2 = \sqrt{2}$, and

$$\delta_j - \delta_{y_i^G} = e_{j,y_i^G}^\top \delta \leq \|e_{j,y_i^G}\|_2 \|\delta\|_2 = \sqrt{2} \|\delta\|_2, \quad (26)$$

where the inequality follows from Cauchy–Schwarz. Combining with the Lipschitz bound,

$$\delta_j - \delta_{y_i^G} \leq \sqrt{2} \sigma_{\max}(W) \|z_{\text{new}} - z_i\|_2. \quad (27)$$

If $\|z_{\text{new}} - z_i\|_2 < m_i/(\sigma_{\max}(W)\sqrt{2})$, then $\delta_j - \delta_{y_i^G} < m_i$, i.e., the sufficient condition holds. Substituting gives $[f(z_{\text{new}})]_{y_i^G} - [f(z_{\text{new}})]_j > 0$ for all $j \neq y_i^G$, and hence y_i^G remains the argmax label at z_{new} . ■

This lemma directly provides the safety criterion for online hits in Section 4: as long as the query feature falls inside the certified ball, the GuardNet classification of that query cannot differ from the cache center—in other words, the cached payload $\tilde{y}_i$ (when set to the proxy label y_i^G) is theoretically reliable.

Corollary 5.1 (Input-level sufficient condition). Given $L_G \leq 1$, any input satisfying $\|x_{\text{new}} - x_i\|_2 < r_i$ is guaranteed to satisfy $\|z_{\text{new}} - z_i\|_2 < r_i$, and hence its GuardNet prediction is y_i^G . More generally, if $L_G > 0$, the sufficient condition is $\|x_{\text{new}} - x_i\|_2 < r_i/L_G$.

Proof: $\|z_{\text{new}} - z_i\|_2 = \|G(x_{\text{new}}) - G(x_i)\|_2 \leq L_G \|x_{\text{new}} - x_i\|_2 < r_i$. The conclusion follows directly from Lemma 5.1. ■

The above corollary shows that, when GuardNet is 1-Lipschitz, the r_i -ball in input space directly constitutes a sufficient region for certified reuse. The online cache decision, however, still uses the feature-space check directly—it is tighter and requires no back-propagation through L_G .

5.3 Tighter Radius and Certificate Scope

Remark 5.1 (Tighter per-pair radius). The $\sqrt{2}$ factor in the radius r_i of Lemma 5.1 comes from taking the worst case over all $C - 1$ competing classes simultaneously. By analyzing each class separately, a tighter radius can be obtained. Let $W_c \in \mathbb{R}^d$ denote the c -th row of W . For each competing class $j \neq y_i^G$, decompose the logit difference as

$$\begin{aligned} [f(z)]_{y_i^G} - [f(z)]_j &= ([f(z_i)]_{y_i^G} - [f(z_i)]_j) \\ &\quad + (W_{y_i^G} - W_j)^\top (z - z_i). \end{aligned} \quad (28)$$

By Cauchy–Schwarz, the cross term is bounded by $-\|W_{y_i^G} - W_j\|_2 \|z - z_i\|_2$, so when $\|z - z_i\|_2 < r_{i,j}$ we have $[f(z)]_{y_i^G} > [f(z)]_j$, where

$$r_{i,j} = \frac{[f(z_i)]_{y_i^G} - [f(z_i)]_j}{\|W_{y_i^G} - W_j\|_2}. \quad (29)$$

If $\|W_{y_i^G} - W_j\|_2 = 0$, the per-pair logit difference is constant and corresponds to no finite constraint. The tight per-sample radius takes the minimum: $r_i^* = \min_{j \neq y_i^G} r_{i,j}$. Since $\|W_{y_i^G} - W_j\|_2 \leq \sqrt{2} \sigma_{\max}(W)$ and $[f(z_i)]_{y_i^G} - [f(z_i)]_j \geq m_i$, we always have $r_{i,j} \geq r_i$, and hence $r_i^* \geq r_i$. For consistency of exposition, the rest of the paper mainly uses the conservative radius r_i ; the experiments report results for both radii.

Remark 5.2 (Certificate scope and capability boundary).

The object certified by Lemma 5.1 is the local consistency of the GuardNet classifier in the feature space: it guarantees that the GuardNet prediction label y_i^G is invariant inside the certified ball. When the cached payload $\tilde{y}_i$ happens to equal y_i^G (the proxy-label mode), the label returned online is directly aligned with the certificate—this is the default setting adopted in the main experiments of this paper.

However, the lemma does *not* provide formal guarantees for the following cases: (1) when the payload source is the MainNet prediction $\hat{y}_M(x_i)$ or the ground-truth label y_i and it is inconsistent with y_i^G , the cached payload $\tilde{y}_i$ returned on a hit may differ from the certified y_i^G ; (2) whether the MainNet prediction over the entire certified ball is consistent with the payload. In other words, the theoretical certificate of LipCache is on the GuardNet side, not on the MainNet side. The consistency between GuardNet and MainNet is indirectly promoted by the training objective and evaluated empirically in the experiments. To obtain a formal MainNet-side guarantee, one may further stack Lipschitz certification or randomized smoothing of MainNet itself.

Combining the derivations of this section, the online cache-hit decision of LipCache obtains a complete theoretical skeleton: starting from the Lipschitz control of GuardNet (Section 4.2), Lemma 5.1 turns the local classification margin into a computable per-sample certified radius, Remark 5.1 provides a larger and less conservative per-sample certified radius, Corollary 5.1 extends the condition to input space, and finally Remark 5.2 clarifies the applicable scope of the certificate. Under this theoretical framework, the experimental evaluation in Section 6 reports not only

the hit rate and accuracy, but also verifies, via the “certified-consistency rate,” whether every accepted cache reuse indeed occurs inside the certified boundary.

6 EXPERIMENTS

Section 5 establishes the theoretical guarantee for cache reuse in `LipCache`: any query that falls inside the certified radius r_i necessarily has a `GuardNet` prediction label consistent with the cache center. This section subjects that theoretical guarantee to empirical examination, organizing the experimental evaluation around three questions. First, on real image classification tasks, can certified caching form non-trivial reuse and stably hold under limited accuracy loss. Second, is the certified radius necessary relative to empirical thresholds—that is, does there exist a simpler alternative boundary that equally satisfies certified consistency. Third, with the certification condition held fixed, how do cache strategy, feature dimensionality, training objective, and class count jointly determine the operating point among hit rate, accuracy, and latency gains. It should be noted that the evaluation goal of this section is the viability of the certification mechanism itself under i.i.d. test distributions, i.e., examining whether the certified radius can form non-trivial reuse while preserving theoretical consistency. Temporal redundancy, concept drift, and consecutive-frame reuse rates in streaming deployment constitute evaluation problems of a different nature, and their benchmark design lies outside the scope of this paper (see Section 7.1).

The main text consistently reports three core metrics: the cache hit rate H (the frequency with which the system reuses the cache), the end-to-end accuracy (the agreement between the overall system output and the ground truth), and the certified-consistency rate (the fraction of all accepted cache hits that satisfy the certification condition; 100% means that no theoretical violation occurred). In addition, the measured system-level speedup is defined as

$$S_{\text{time}} = \frac{T_{\text{Main}}}{T_{\text{guard}} + T_{\text{search}} + T_{\text{fallback}}},$$

where the numerator is the measured time of executing `MainNet` alone, and the denominator is the measured total time of `GuardNet` encoding, cache search, and, on a miss, the `MainNet` fallback. Although the speedup is sometimes written as $1/(1 - H)$, this expression only reflects the reduction in `MainNet` invocations and ignores the encoding and search overhead; we therefore adopt the measured definition S_{time} above as the reported metric.

6.1 Experimental Setup

To keep the experiments comparable, the main-text experiments share a common base protocol. `MainNet` is responsible only for fallback inference on misses and serves as the standalone-accuracy and timing baseline; `GuardNet` is responsible for extracting features that satisfy the 1-Lipschitz constraint and accordingly defining the theoretical radius for cache reuse. Unless stated otherwise, the three 10-class tasks all use the tight certified radius, a cache-construction method that prioritizes the certified radius combined with de-duplication, returns the proxy label on a hit, caches 200 samples per class, and repeats the experiments over random

seeds {42, 123, 2024}. The standalone accuracies of the three `MainNet` models are as follows: 94.78% on CIFAR-10 [34], 87.0% on Tiny-ImageNet 10 classes (denoted Tiny-10) [35], and 97.19% on SVHN [36]. In the current final experiments, the certified-consistency rate of all certified-radius results is 100%, and no Lipschitz-constraint violation or theoretical violation at the proxy-decision level is observed.

The training-objective ablation changes only how `GuardNet` is trained, with everything else held fixed; the multi-class extension experiment changes only the class count and the corresponding `GuardNet` training recipe; the perturbation and deployment experiments examine the behavior of the system under distributional perturbation and latency constraints on top of the default operating point.

Scope of the evaluation: The experiments are designed to validate the viability and cross-task stability of the per-sample certified reuse mechanism, rather than to propose a new SOTA image classification benchmark. The three tasks respectively represent three typical 10-class distributions—natural images, fine-grained classification, and domain shift—which suffice to examine the sensitivity of the certified radius to task geometry. Extension to full ImageNet (1000 classes), CIFAR-100, and streaming video benchmarks lies outside the scope of this paper and is left to future work.

6.2 Feasibility of Certified Caching

Figure 3 reports the hit rate, end-to-end accuracy, certified-consistency rate, and measured speedup on the three 10-class tasks. Figure 4 portrays the three tasks side by side from the perspective of relative operating points.

Under the conservative certified radius, the average hit rates of CIFAR-10, Tiny-10, and SVHN are 0.341 ± 0.010 , 0.303 ± 0.013 , and 0.470 ± 0.007 , respectively; switching to the tight certified radius, the hit rates rise to 0.342 ± 0.011 , 0.472 ± 0.012 , and 0.502 ± 0.013 . The corresponding end-to-end accuracies remain at 0.921, 0.867, and 0.960, and all results pass the certified-consistency check.

A non-trivial certified reuse region is observed on all three tasks. Under the tight certified radius, the hit rates of CIFAR-10, Tiny-10, and SVHN are all significantly above zero, indicating that `LipCache` does not trigger theoretically safe reuse only in the vicinity of a tiny minority of samples, but rather can form stable usable hits over the full test set. Meanwhile, the end-to-end accuracy is only about 2.70, 0.27, and 1.19 percentage points lower than the respective standalone `MainNet` accuracy, showing that certified caching does not trade a large loss in prediction quality for reuse opportunities.

The difference between the conservative and the tight certified radius reveals the basic principle that “the certification boundary can be tightened but cannot be loosened arbitrarily.” On CIFAR-10, the two radii are already close, and the tight radius brings only a 0.15 percentage-point gain in hit rate and about $0.04\times$ in speedup, with accuracy nearly unchanged. On Tiny-10, the tight certified radius raises the hit rate from 0.303 to 0.472 and the measured speedup from $1.11\times$ to $1.31\times$, with accuracy dropping by only about 0.20 percentage points—the conservative radius is overly cautious on higher-resolution images. SVHN lies between the two: the hit rate rises from 0.470 to 0.502, the speedup

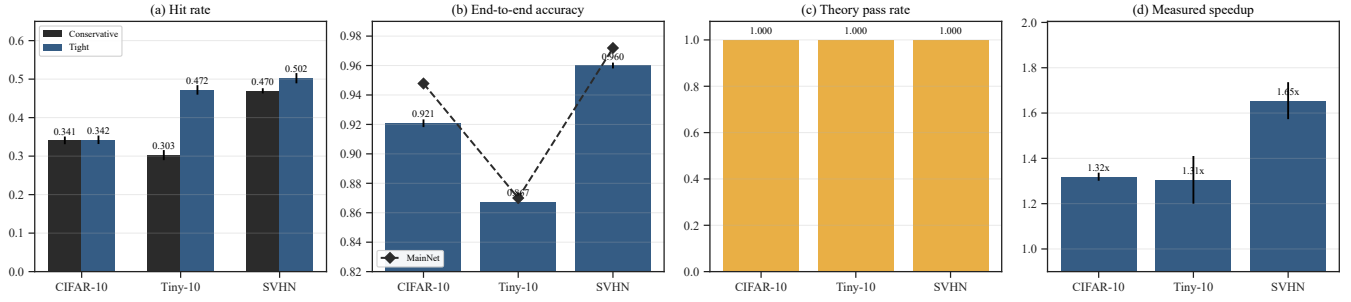


Fig. 3. Main results on the three 10-class tasks. From left to right: cache hit rates under the conservative versus tight certified radius; end-to-end accuracy under the tight radius, with the standalone *MainNet* accuracy marked by a dashed line; certified-consistency rate; and measured speedup. All certified-radius results pass the consistency check.

from $1.58\times$ to $1.65\times$, and the accuracy drops by about 0.21 percentage points. Overall, the tight certified radius trades a moderately enlarged certified reuse region for a higher hit rate, but without breaking certified consistency.

The differences across the three datasets reflect the systematic influence of task geometry on the efficiency of certified reuse. SVHN simultaneously exhibits the highest hit rate and the highest measured speedup—on tasks such as digit images, where intra-class variation is constrained, and the local geometry learned by *GuardNet* is more stable, and the certified reuse region more easily covers test samples. The hit rate of Tiny-10 is notably higher than that of CIFAR-10, but its speed gain does not increase proportionally, because the higher input resolution raises the cost of *GuardNet* encoding and nearest-neighbor search, weakening the translation of hit-rate growth into real wall-clock gains. CIFAR-10 has the lowest hit rate, yet it still maintains a measured speedup above $1.3\times$ —under a lower encoding cost, medium-scale certified hits are already sufficient to yield stable system gains.

Figure 4 displays, in a column-normalized side-by-side manner, the hit rate under the tight certified radius, the hit-rate gain relative to the conservative radius, the accuracy retention relative to *MainNet*, the overall prediction agreement with *MainNet*, and the measured speedup. SVHN dominates simultaneously on hit rate, accuracy retention, and speedup; Tiny-10 stands out most on hit-rate gain; and CIFAR-10 maintains a high level of agreement with *MainNet* but has a limited hit-rate gain. This structured comparison further shows that the different benefit dimensions of *LipCache* are emphasized differently across tasks.

In terms of random-seed stability, the fluctuations of the three main results are all small. Under the tight certified radius, the hit-rate standard deviations of CIFAR-10, Tiny-10, and SVHN are 0.011, 0.012, and 0.013, respectively, and the accuracy standard deviations all remain at about the 0.002 level or below, indicating that the main conclusions do not depend on a particular training seed or cache-sample composition.

Overall, Figures 3 and 4 jointly establish a consistent result pattern for *LipCache* on three 10-class tasks with markedly different statistical structure: non-zero certified hits, limited accuracy loss, measurable latency gains, and zero theoretical violations.

6.3 Comparison with Related Efficiency Systems

Related work shares the goal of reducing inference cost, but the savings occur at different points in the execution path. BranchyNet [7] and MSDNet [19] retrain multi-exit networks so that easy inputs leave early at the execution node; they reduce computation after a request reaches that node and do not reuse results associated with prior inputs. NoScope [9] specializes a cascade to fixed-camera video, DeepCache [10] reuses regions across consecutive video frames, and Semantic Memory [11] applies empirical semantic caching through cross-layer similarity. Their locality assumptions, acceptance conditions, and deployment paths differ from ours. *LipCache* keeps *MainNet* unchanged, uses a lightweight *GuardNet* to retrieve cached proxy labels at the edge, and contacts the cloud only on a miss. It therefore first reduces uploads and cloud invocations, and can then be composed with a cloud early-exit network to reduce the cloud computation of misses.

6.3.1 Edge–Cloud Resource Accounting

Table 2 reports both edge-resident model footprint and a common edge–cloud deployment accounting model. Parameter counts and raw FP32 weights describe static weights in the current CIFAR-10 implementations, excluding activations, runtime state, cache entries, and framework overhead; they are therefore necessary but not sufficient evidence of edge-deployment feasibility. If an early-exit network is placed at the edge, its complete backbone and all exits must remain resident: the three-exit BranchyNet has 1.759M parameters/6.71 MiB and the seven-exit MSDNet has 2.987M/11.39 MiB, whereas the base *GuardNet* ($d = 64, c = 32$) used by *LipCache* has 0.282M/1.08 MiB. Normalize the compute budget of a complete cloud prediction to 100%, let the certified CIFAR-10 operating point have hit rate $H = 0.348$, and use the native early-exit compute speedups relative to each system’s complete backend, $S_B = 2.31$ for BranchyNet and $S_M = 2.14$ for MSDNet. For *LipCache*, the uploaded-request fraction is $1 - H$; when a cloud early-exit backend handles only misses, normalized cloud computation is $(1 - H)/S_B$ or $(1 - H)/S_M$. Cloud-compute saving is $1 - C$ and cloud-capacity gain is $1/C$, where C is the normalized cloud computation.

The accounting exposes the complementarity. Cloud BranchyNet and MSDNet reduce cloud computation to

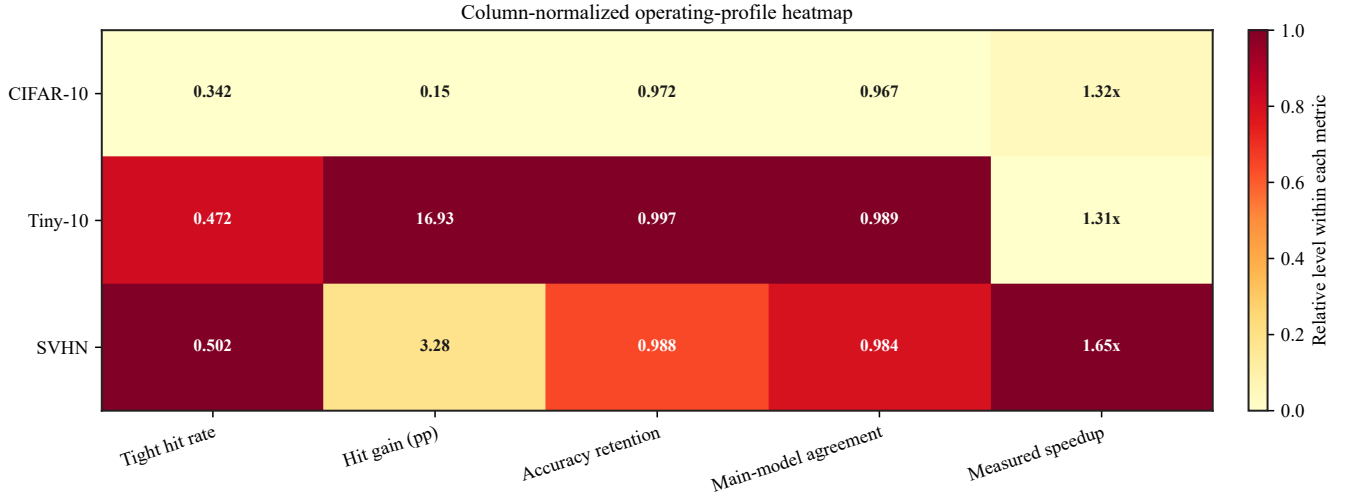


Fig. 4. Operating-point profile heatmap of the three 10-class tasks. The columns correspond to the hit rate under the tight certified radius, the hit-rate gain relative to the conservative radius, the accuracy retention relative to `MainNet`, the overall prediction agreement with `MainNet`, and the measured speedup. Colors are normalized within each column; the annotated numbers retain their original magnitudes.

TABLE 2

Edge-resident model footprint and edge–cloud resource accounting at the CIFAR-10 operating point. Footprint denotes the static FP32 weight volume of the edge-resident model, distinct from the runtime compute cost analyzed in later subsections. Uplink traffic and cloud computation are normalized to a complete cloud prediction (100%); cloud computation applies early-exit speedups as conditional-compute multipliers, yielding a deployment accounting model rather than a cross-architecture comparison of FLOPs, latency, accuracy, or energy.

ID	Configuration (edge → cloud)	Edge footprint (params/size in FP32 MiB)	Uplink traffic (vs. all-cloud)	Cloud compute	Compute saved	Capacity gain
<i>Cloud-only baselines</i>						
(1)	— → <code>MainNet</code>	—	100.0%	100.0%	0.0%	1.00×
(2)	— → <code>BranchyNet</code>	1.759M [†] / 6.71 [†]	100.0%	43.3%	56.7%	2.31×
(3)	— → <code>MSDNet</code>	2.987M [†] / 11.39 [†]	100.0%	46.7%	53.3%	2.14×
<i>Edge LipCache + cloud backend</i>						
(4)	<code>GuardNet</code> → <code>MainNet</code>	0.282M / 1.08	65.2%	65.2%	34.8%	1.53×
(5)	<code>GuardNet</code> → <code>BranchyNet</code>	0.282M / 1.08	65.2%	28.2%	71.8%	3.54×
(6)	<code>GuardNet</code> → <code>MSDNet</code>	0.282M / 1.08	65.2%	30.5%	69.5%	3.28×

[†]: footprint if the cloud-resident network were deployed at the edge (shown for size comparison only).

43.3% and 46.7%, respectively, but leave uploads at 100%. `LipCache` first hits at the edge for 34.8% of requests, reducing both uploads and cloud invocations to 65.2%. In composition, cloud computation further falls to 28.2% or 30.5%, corresponding to 3.54× or 3.28× cloud-capacity gain. The $H = 0.348$ value is the controlled CIFAR-10 operating point of this section and differs from the $H = 0.3295$ real-device path in Section 6.8; the two are not mixed. Early-exit networks are not restricted to cloud deployment; the table describes their resource effect when used as the cloud backend. Certification still covers only local `GuardNet` proxy-label consistency when a cached label is returned, not the early-exit network, `MainNet`, or ground-truth correctness. Combined end-to-end latency, accuracy, edge energy, and live network traffic require measurement on a unified path.

6.4 Certified Radius and Empirical Thresholds

Figure 5 compares three kinds of reuse boundaries: the tight certified radius, an empirical radius based on inter-class distance statistics, and a single global nearest-neighbor threshold. The three datasets exhibit a consistent pattern: the certified radius does not pursue the highest hit rate, but

it is the only boundary definition that can stably keep the certified-consistency rate at 100% across all three tasks.

Taking CIFAR-10 as an example, the hit rate, accuracy, and consistency corresponding to the tight certified radius are 0.344, 0.922, and 100%, respectively; the inter-class-distance empirical radius slightly raises the hit rate to 0.352, but the certified-consistency rate drops to 37.9%; the global nearest-neighbor threshold pushes the hit rate up to 0.494, yet it drives the end-to-end accuracy down to 0.777. On Tiny-10, the empirical thresholds are likewise more aggressive, but their corresponding certified-consistency rates are only 33.3% and 23.9%; on SVHN, the empirical thresholds do not even achieve a higher hit rate, yet they still depress the certified-consistency rate to 45.7% and 25.4%. The empirical thresholds either trade accuracy and consistency for a higher hit rate, or damage the certification condition without notably improving the hit rate; they do not form a better accuracy–hit-rate frontier across the three tasks.

This result defines the role of the certified radius in the main text: it does not pursue the most aggressive empirical hit rate, but provides a reproducible, interpretable, and cross-task consistent reuse boundary.

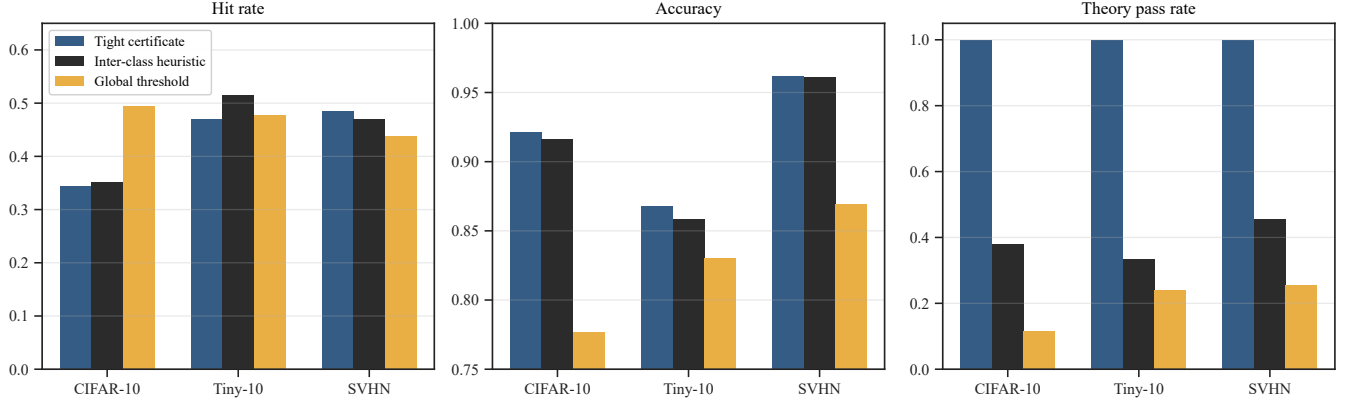


Fig. 5. Comparison of the certified radius and empirical thresholds. Empirical thresholds sometimes yield a higher hit rate, but only the tight certified radius keeps the certified-consistency rate at 100% across all three 10-class tasks while maintaining high end-to-end accuracy.

6.5 Default Operating Point Configuration

Figures 6 and 7 answer why the main-text default configuration adopts the operating point of “tight certified radius + radius-prioritized de-duplication + proxy label + 64-dimensional features + 200 samples per class.”

On the entry-selection strategy, radius-prioritized de-duplication gives the highest hit rate on CIFAR-10 and also outperforms random sampling and K-center on Tiny-10; on SVHN, the hit rate of K-means is slightly higher than that of radius-prioritized de-duplication, but the gap is small. Considering the consistency and interpretability across the three tasks, radius-prioritized de-duplication constitutes a more robust unified default strategy, which also indicates that the certified-caching scenario requires not only geometric coverage but also prioritized retention of high-quality center samples with large certified radii.

Regarding return-value semantics, the proxy label is directly aligned with the theoretical guarantee and provides the clearest semantics. On CIFAR-10, the end-to-end accuracies of the proxy-label, MainNet-label, and ground-truth schemes are 0.9215, 0.9134, and 0.9134, respectively; on Tiny-10, they are 0.8680, 0.8680, and 0.8700, respectively; and on SVHN, they are 0.9617, 0.9241, and 0.9237, respectively. These results show that the proxy label achieves higher end-to-end accuracy on CIFAR-10 and SVHN, whereas the differences among the three semantics are small on Tiny-10. In all cases, the proxy label remains fully aligned with the certification condition. The main text adopts the proxy label as the default precisely because it has the clearest theoretical semantics under a unified protocol.

Cache capacity remains an effective lever. On all three datasets, increasing the cache from 100 to 400 samples per class raises the hit rate: CIFAR-10 from 0.288 to 0.392, Tiny-10 from 0.446 to 0.480, and SVHN from 0.434 to 0.531. Capacity growth, however, does not automatically bring better accuracy or lower latency; the final operating point should therefore be selected jointly with the radius-tightening strategy discussed below.

The optimal feature dimensionality is dataset-dependent. On CIFAR-10, the hit rates for $d = 32/64/128$ are 0.348/0.344/0.336, respectively, and 32 dimensions are already near-optimal; on Tiny-10, the corresponding hit

rates are 0.456/0.470/0.452, with little difference between 32 and 64 dimensions; on SVHN, the hit rate continues to rise from 0.498/0.484 to 0.527, so higher dimensions still bring gains. The main text retains 64 dimensions as the unified default in order to share the same protocol across the three tasks; the optimal dimensionality of different datasets is not fully consistent, and this is precisely the degree of freedom that needs to be re-tuned per task in subsequent deployment.

6.6 Hit Mechanism and Radius Tightening

After determining the default operating point, two phenomena still need to be explained: why incorrect hits occur mainly near the boundary, and why tightening the radius can improve accuracy without breaking the certification condition. Figure 8 provides the answer.

On all three tasks, the normalized distance ratio $\|z_q - z_i\|_2/r_i$ of correct hits is consistently and markedly lower than that of incorrect hits. The mean distance ratios of correct hits on CIFAR-10, Tiny-10, and SVHN are 0.802, 0.752, and 0.767, respectively, whereas those of incorrect hits rise to 0.891, 0.910, and 0.908, systematically closer to the certification boundary. Meanwhile, the cache-center classification margin of correct hits is significantly larger: 2.43, 3.77, and 2.38 on the three datasets, versus only 1.63, 1.97, and 1.49 for incorrect hits. This indicates that high-risk hits do not appear at random, but are concentrated in the region “closer to the boundary and with a smaller classification margin.”

The lower panels of Figure 8 show the radius-tightening phenomenon directly. When the radius is progressively tightened on top of the conservative certified radius, all three datasets exhibit a decrease in hit rate and an increase in accuracy, while the certified-consistency rate remains at 100% throughout. For example, on CIFAR-10 the hit rate drops from 0.344 to 0.200 and the accuracy rises from 0.922 to 0.940; on SVHN the hit rate drops from 0.460 to 0.312 and the accuracy rises from 0.964 to 0.970. Radius tightening does not change whether the certification holds; rather, by removing high-risk hits closer to the boundary, it pushes the system toward a more conservative but higher-accuracy operating point.

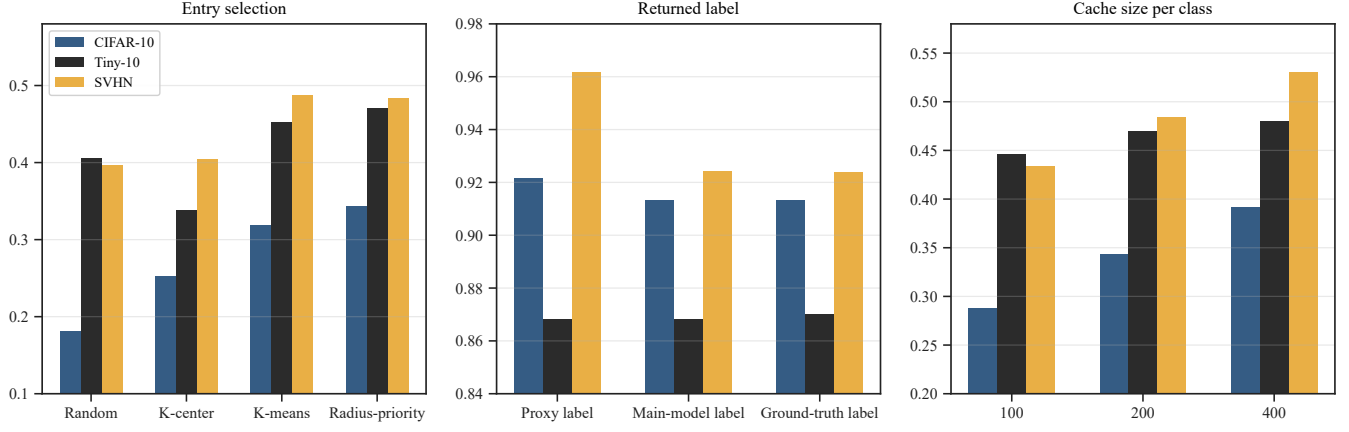


Fig. 6. Default-operating-point ablation on the three 10-class tasks. Left: effect of entry-selection strategy on the hit rate; middle: effect of return-value semantics on end-to-end accuracy; right: effect of per-class cache capacity on the hit rate.

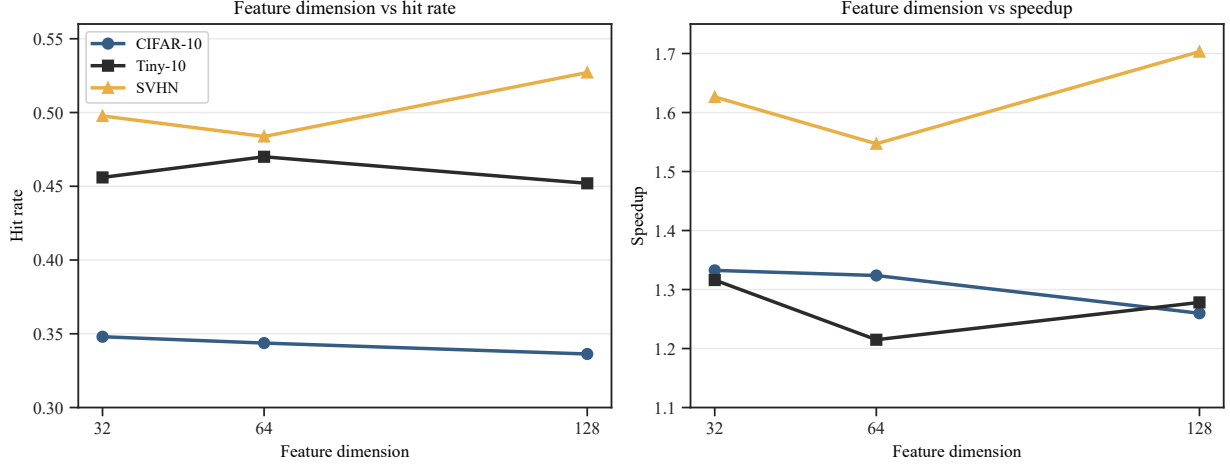


Fig. 7. Effect of feature dimensionality on the hit rate and the measured speedup. CIFAR-10 and Tiny-10 are already near their respective optima at smaller dimensions, whereas SVHN continues to improve at higher dimensions, indicating that the unified default dimensionality is not fully identical to each dataset's optimum.

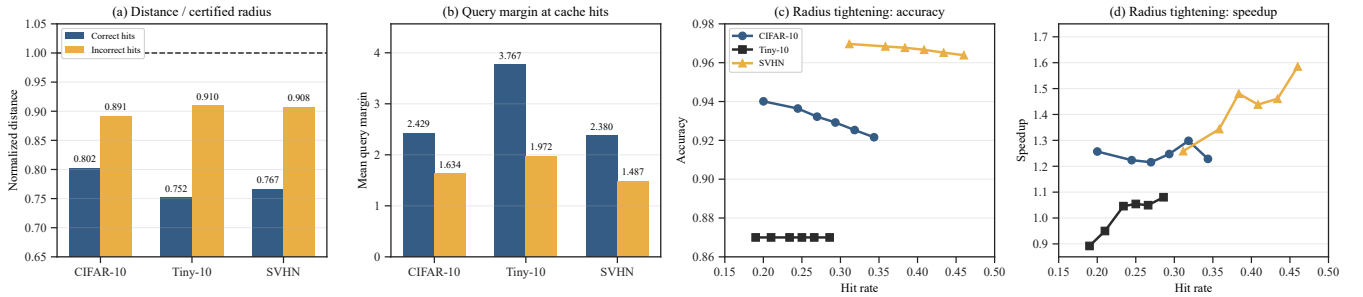


Fig. 8. Hit mechanism and radius tightening. Top: incorrect hits are closer to the certified boundary $\|z_q - z_i\|_2 = r_i$ and have a smaller cache-hit margin. Bottom: progressively tightening the certified radius lowers the hit rate while raising accuracy and preserving theoretical self-consistency on all three 10-class tasks.

6.7 Training Objective and Multi-Class Extension

The above results show that the main degrees of freedom of the caching system lie not in whether to certify, but in how to change the hit-rate–accuracy operating point through training and radius selection. We therefore examine, respectively, the training-objective ablation on CIFAR-10 and the multi-class extension experiment on Tiny-ImageNet.

Figure 9 gives the training-objective ablation results on CIFAR-10. Cross-entropy supervision alone provides the most robust default operating point among the current 10-class main experiments, with a hit rate and accuracy of 0.358 and 0.920, respectively. The margin-enlargement regularizer mildly raises the hit rate: under a stronger margin-regularization setting, the hit rate rises to 0.401 and the accuracy drops to 0.909. The center constraint that compresses the intra-class distribution is more aggressive: the hit rate can be further raised to 0.544, 0.596, and even 0.644, with the corresponding accuracy dropping to 0.894, 0.888, and 0.877. It is worth emphasizing that all training-objective variants pass the certified-consistency check—the training objective changes the operating point, not the theoretical validity itself.

Figure 10 answers a more critical question: after a stronger GuardNet training recipe, can certified caching scale from 10 classes to larger class counts? On the cross-entropy-only multi-class baseline, the hit rates under the tight certified radius at 20/30/50 classes are only 0.056, 0.0047, and 0.0004, respectively; after adopting the enhanced training recipe, these three values rise to 0.423 ± 0.009 , 0.254 ± 0.004 , and 0.124 ± 0.001 , respectively, while the end-to-end accuracy remains at 0.838, 0.847, and 0.836, and the certified-consistency rate stays at 100%. This indicates that the main bottleneck in the multi-class scenario lies not in the caching mechanism itself, but in whether GuardNet can learn a sufficiently compact and separable certified feature geometry.

Nevertheless, the growth in class count remains the dominant factor determining the upper bound of reusability. Even with the enhanced training recipe, the hit rate keeps declining as the class count increases. Increasing the feature dimensionality cannot fundamentally reverse this trend: under the 20/30/50-class settings, $d = 32$ achieves the highest hit rates of 0.431, 0.266, and 0.158, respectively, all outperforming $d = 64$ under the same settings. The message conveyed by the multi-class extension is not “one can abandon certification when the class count grows,” but rather “improving GuardNet training can markedly enlarge the usable class scale while preserving the certification condition, yet the class count still determines the upper bound of certified reuse.”

6.8 Query Perturbation and Deployment Boundaries

Figure 11 gives the empirical stress-test results under query perturbation. This group of experiments should be understood as an empirical generalization assessment rather than a direct extrapolation of the theoretical certificate. All three datasets exhibit the same pattern: additive noise is the most stable, JPEG compression brings a mild decline, and cropping and scaling are the most damaging to the hit rate. For example, on CIFAR-10 the baseline hit rate under the

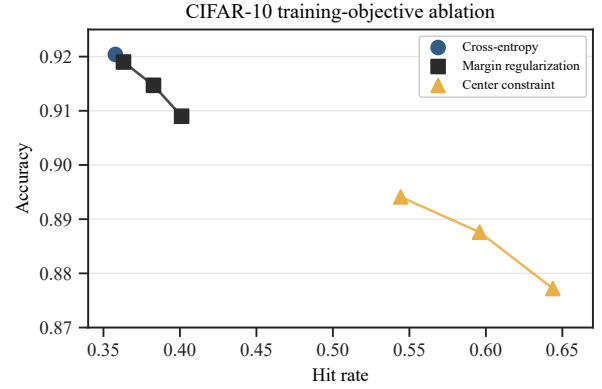


Fig. 9. Training-objective ablation on CIFAR-10. Cross-entropy supervision gives the most robust default operating point; margin regularization can mildly raise the hit rate; the center constraint can further raise the hit rate, but at a more noticeable accuracy cost.

tight certified radius is 0.343; it drops to 0.310 under strong JPEG perturbation, and drops to 0.093 and 0.152 under strong cropping and strong scaling, respectively. Tiny-10 and SVHN preserve the same trend.

More importantly, the performance degradation under perturbation comes mainly from “not hitting” rather than “hitting but being wrong.” In all perturbation experiments, the certified-consistency rate remains at 100%, indicating that certified consistency itself is not broken; the performance change is mainly because the perturbed query points more easily escape the original certified ball, rather than because local consistency inside the certified ball has failed.

It should be emphasized that the above perturbation tests examine the robustness of the certified radius to single-point local perturbations and remain within the i.i.d. evaluation framework. Temporal redundancy (e.g., inter-frame coherence in video) and concept drift in streaming scenarios involve sequential correlations and require dedicated streaming benchmarks together with an analysis of how the hit rate evolves over time; these are left as future work. The results in this section can serve as an upper-bound reference for streaming deployment: if the i.i.d. hit rate is H_0 , then under strong temporal correlation the probability that consecutive frames fall inside the same certified ball is typically no lower than H_0 .

We further deploy the baked CIFAR-10 GuardNet on an Atlas 200I DK A2 equipped with a DaVinci 310B4 NPU. The edge board executes GuardNet and cache lookup, while the remote MainNet latency is measured with ResNet50 [37] on an RTX 4070 Ti. The device evaluation uses 2,000 queries and a 2,000-entry cache, obtaining a hit rate of 0.3295 and an end-to-end accuracy of 0.9170.

Figure 12 reports the real-device latency components and the RTT-based split-inference estimate. The NPU takes 1.34ms/query for GuardNet encoding and the cKDTREE lookup takes 0.22ms/query, giving an edge-local path of 1.56ms/query. Cloud batch-1 MainNet inference takes 8.38ms/query. Combining these measured components with LAN, WiFi, and cellular RTTs of 2/10/40ms gives expected speedups of 1.22 $\times$, 1.32 $\times$, and 1.42 $\times$, respectively.

Figure 13 exposes two limits that are hidden by the

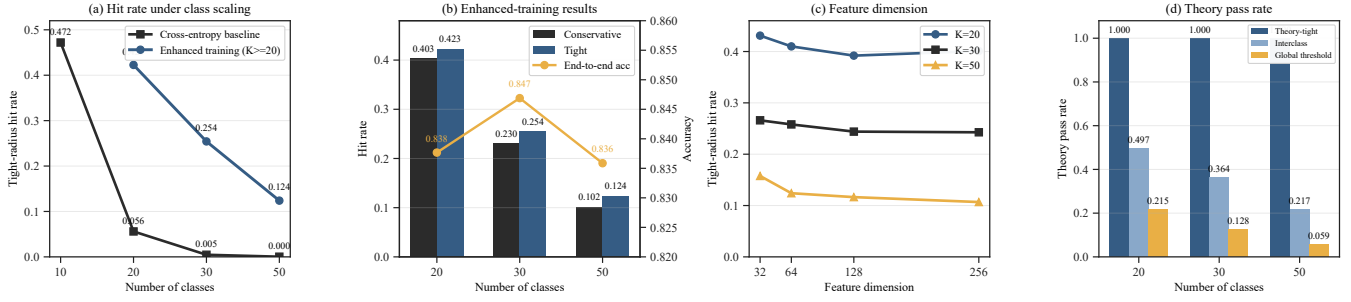


Fig. 10. Tiny-ImageNet multi-class extension experiment. Top left: comparison of the tight-certified-radius hit rates of the cross-entropy-only baseline and the enhanced training recipe at different class counts; top right: conservative versus tight certified-radius hit rates of the enhanced training recipe at 20/30/50 classes and the corresponding end-to-end accuracy; bottom left: relationship between feature dimensionality and hit rate at different class scales; bottom right: certification consistency of different radius definitions in the multi-class scenario.

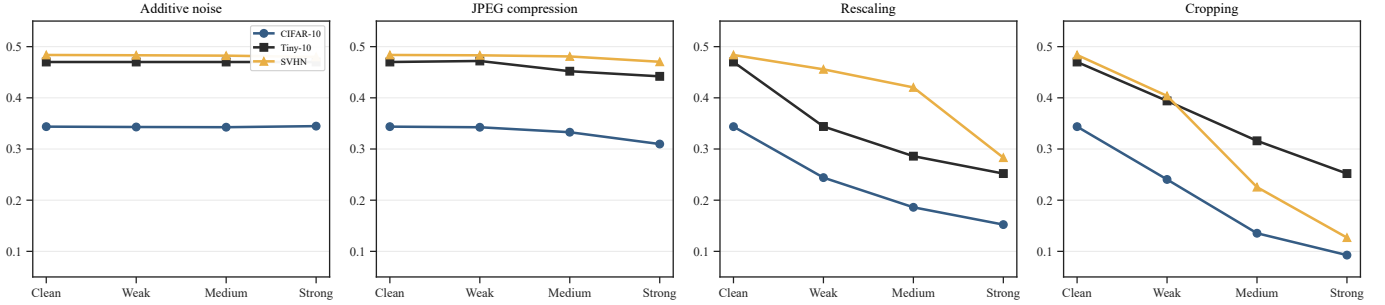


Fig. 11. Empirical stress test under query perturbation. Noise perturbation barely affects the certified hit rate of the three 10-class tasks; JPEG compression brings a mild decline; cropping and scaling more easily push query points out of the certified ball, so the hit-rate decline is more pronounced.

batch-1 result. First, the cloud GPU reduces its per-image time from 8.38ms at batch 1 to 0.207ms at batch 128, whereas the edge NPU encoding time follows a shallow U-shape (1.34/0.81/0.87/1.18 ms). Second, a discrete-event simulation driven by these measured per-operation times shows a saturation boundary: at 1,000 queries/s, the edge reaches approximately 99% utilization and *LipCache* p99 latency rises to 177.4ms, compared with 27.8ms for all-cloud inference. Thus, edge caching is advantageous when RTT is material and the edge retains spare capacity, but it can move the bottleneck from the cloud to the edge under extreme load.

The measurement scope is deliberately separated in the figures. NPU encoding, cache lookup, and cloud inference are hardware measurements; the network speedups combine those measurements with modeled RTT rather than a live HTTP round trip; and the load experiment is a discrete-event simulation parameterized by measured operation times rather than a real traffic trace.

6.9 Retrieval Complexity Analysis

Figure 14 isolates the variable retrieval term in Eq. (16). We fix C_G and $\text{InvokeRate}_M C_M$ at the final CIFAR-10 seed-42 operating point ($d = 64$, $N = 2,000$, hit rate 0.3437, hence $\text{InvokeRate}_M = 0.6563$). The left panel fixes $N = 2,000$ and varies d ; the right fixes $d = 64$ and varies N . The unchanged additive terms are normalized to one work unit, so both panels directly show the linear contribution of Nd . This is an analytical complexity analysis of the online decision rule, not a hardware-latency measurement.

6.10 Section Conclusions

The experiments in this section support the following conclusions. First, *LipCache* can stably achieve certified cache reuse on the three 10-class tasks of CIFAR-10, Tiny-10, and SVHN, and the main results consistently pass the certified-consistency check. Second, although the certified radius is more conservative than empirical thresholds, it is the only reuse boundary that can stably preserve certified consistency across tasks. Third, cache strategy, return-value semantics, feature dimensionality, and radius tightening mainly determine the operating point among hit rate, accuracy, and latency, rather than whether certification holds. Fourth, the training-objective ablation on CIFAR-10 and the multi-class extension on Tiny-ImageNet jointly show that improving *GuardNet* training can markedly change the operating point and even enlarge the usable class scale, yet the growth in class count remains the core factor limiting the upper bound of certified reuse. Finally, the perturbation, deployment, and complexity analyses show that the applicability domain of *LipCache* is jointly determined by the local feature geometry, cache scale, the edge encoding cost, and the network RTT, providing clear boundary conditions for subsequent system-level extensions.

7 CONCLUSION AND FUTURE WORK

This paper proposed *LipCache*, a certified semantic caching framework for resource-constrained edge image classification. *LipCache* retains the high-accuracy *MainNet* as a fallback predictor and introduces a

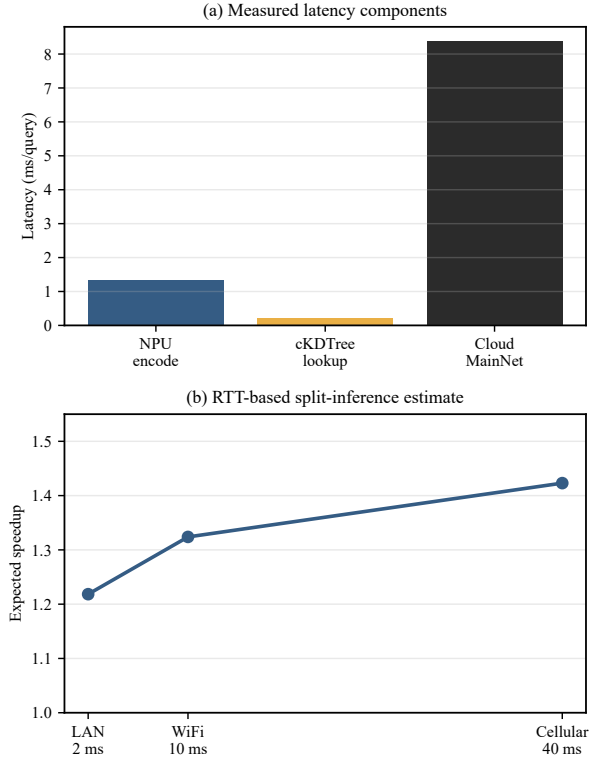


Fig. 12. Real-device measurements for the CIFAR-10 operating point. Left: measured latency components on the DaVinci 310B4 edge NPU and RTX 4070 Ti cloud GPU. Right: expected speedup obtained by combining the measured components with modeled LAN, WiFi, and cellular RTTs. The right panel is an analytical estimate, not a measured HTTP round trip.

lightweight, Lipschitz-controlled *GuardNet* that makes cache-hit decisions in a compact feature space. By deriving a per-sample certified reuse radius from the local *GuardNet* margin and the Lipschitz bound of the classification head, the framework turns cache lookup into an interpretable local-consistency check, rather than an exact-match or global heuristic-threshold decision.

The experimental results show that, on three 10-class tasks with markedly different statistical structure, *LipCache* consistently exhibits the following pattern of results: “non-zero certified hits, limited accuracy loss, measurable latency gains, and zero theoretical violations.” The comparison between the certified radius and empirical thresholds further shows that the per-sample safety criterion does not pursue the most aggressive hit rate, but provides the only reuse boundary that can stably preserve certified consistency across tasks. Entry-selection strategy, return-value semantics, feature dimensionality, and radius tightening mainly affect the operating point among hit rate, accuracy, and latency, rather than whether certification holds. The multi-class extension experiment further reveals that the main limitation of the current method lies not in the cache criterion itself, but in whether *GuardNet* can learn a sufficiently compact and separable certified feature geometry. The edge-deployment analysis further indicates that the system gain of *LipCache* is jointly determined by the hit rate, the guard-side encoding cost, and the network RTT, and that it is best suited to deployment scenarios where

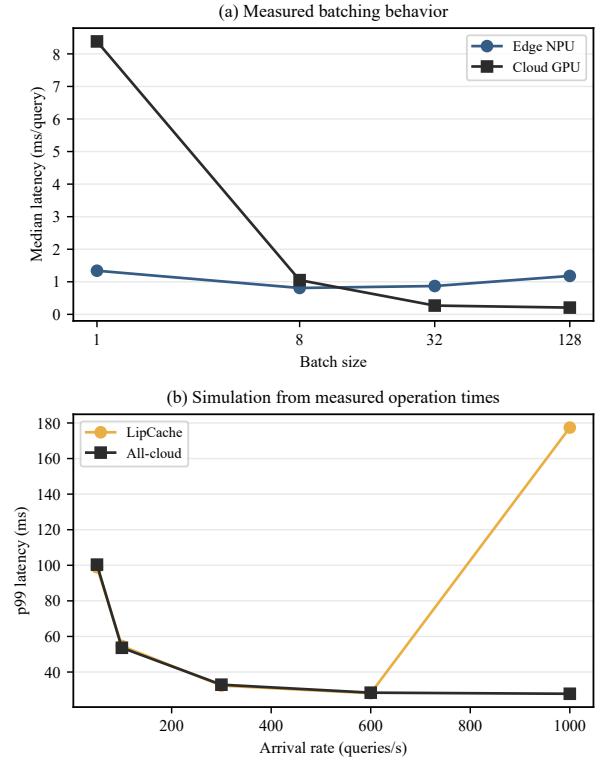


Fig. 13. Operating boundary of the edge deployment. Left: measured per-query latency under different batch sizes, showing the much stronger batching benefit of the cloud GPU. Right: p99 latency from a discrete-event simulation parameterized by measured operation times; the sharp rise near 1,000 queries/s marks edge saturation.

Eq. (16): fixed C_G and $\text{InvokeRate}_M C_M$ ($\text{InvokeRate}_M = 0.6563$)

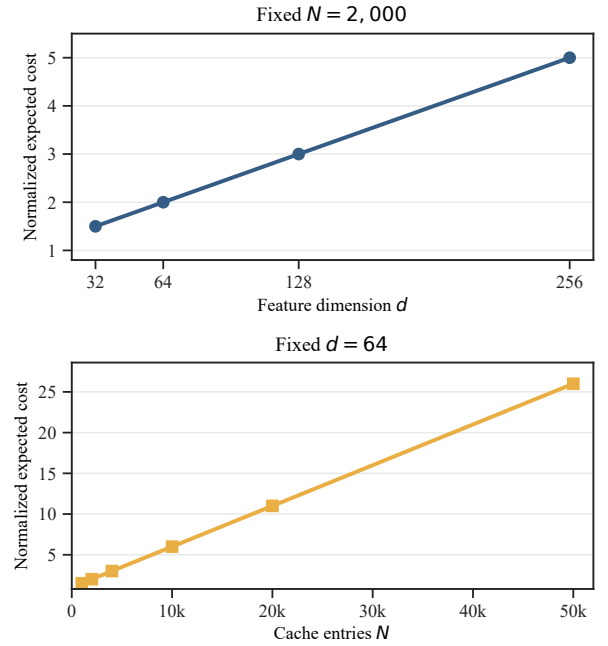


Fig. 14. Complexity analysis corresponding to Eq. (16). C_G and $\text{InvokeRate}_M C_M$ are fixed at the final CIFAR-10 seed-42 operating point; the unchanged additive terms are normalized to one work unit. Left: with $N = 2,000$ fixed, the expected cost grows linearly with d . Right: with $d = 64$ fixed, it grows linearly with N .

“cloud invocations are expensive while the edge side can afford a lightweight proxy.”

In summary, *LipCache* offers a feasible path toward reliable cache-assisted inference at the edge, supported by theoretical analysis and experiments across multiple tasks. The remainder of this section discusses a number of open questions and extension directions along this path.

7.1 Future Work

Before outlining future work, we first clarify the scope and boundaries of the current study, so that subsequent research can advance in a targeted manner.

Task scale and type. The experiments in this paper validate the viability of the certification mechanism on three 10-class image classification tasks, designed to isolate the sensitivity of the certified radius to task geometry. Extension to larger-scale benchmarks such as full ImageNet (1000 classes) and CIFAR-100 is a natural next step.

Evaluation distribution. All experiments are conducted on i.i.d. test sets, targeting a controlled validation of the certification mechanism itself. Temporal redundancy, concept drift, and consecutive-frame reuse rates in streaming deployment involve sequential correlations and require dedicated benchmarks together with a temporal analysis of the hit rate. The perturbation experiments in Section 6.8 can serve as an upper-bound reference for streaming deployment, but a full temporal validation is left as future work.

Energy boundary. The goal of this paper is cache-reuse efficiency and inference acceleration, rather than end-to-end energy reduction. Reducing the number of *MainNet* invocations mechanically reduces energy consumption, but a complete energy assessment would require hardware-in-the-loop measurement, which lies outside the methodological scope of this paper.

Given these boundaries, although *LipCache* already achieves a good balance among accuracy, latency, and throughput, the current design still centers on cache-reuse efficiency and inference acceleration, rather than end-to-end energy reduction. For resource-constrained edge deployment, reducing the number of main-model invocations is only part of the problem. A natural next step is therefore to introduce a spiking neural network (SNN) variant of *GuardNet* on the guard side, so that the system can further reduce energy consumption while preserving discriminative performance and certified cache-reuse validity. This subsection first reports an already-implemented exploratory SNN prototype and its preliminary observations, and then, building on it, discusses broader open directions.

7.2 Exploration: An SNN-Based GuardNet

The above discussion is not purely speculative. An SNN-based *GuardNet* variant has been implemented to evaluate its structural compatibility with the *LipCache* pipeline. The current design retains the same teacher-student training philosophy as the ANN *GuardNet*, but replaces static activations with multi-step spiking dynamics, so that the guard model may benefit from event-driven computation in future low-power deployments.

Figure 15 summarizes the architecture of the current SNN-based *GuardNet*. Each stage follows a “convolution–batch normalization–LIF neuron–average pooling” pattern; higher-layer feature maps are further projected through linear feature layers and additional LIF units, then averaged over time and mapped back into a bounded feature space. The implementation explicitly exposes the number of time steps T , the membrane constant τ , the input scaling, and an optional Poisson-style input encoding, making the model suitable for further analysis of temporal-encoding behavior.

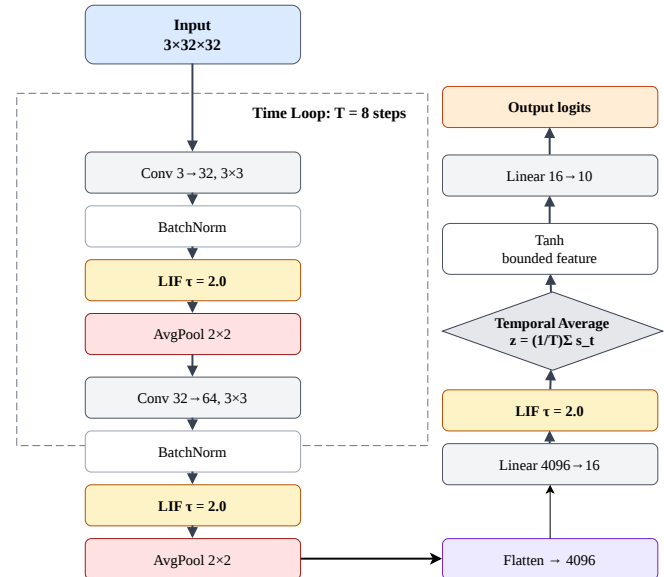


Fig. 15. Architecture of the current SNN-based *GuardNet*. The model replaces the static activations of the original ANN *GuardNet* with multi-step spiking dynamics, while still producing compact features for *GuardNet*-side classification and future cache lookup.

Preliminary observations on CIFAR-10 show that the SNN-based *GuardNet* is trainable and inherits part of the semantic structure learned by the ANN teacher, but they also clearly indicate that the current version is not yet sufficient to replace the ANN guard in the main *LipCache* pipeline. Specifically, the SNN-based *GuardNet* reaches a classification accuracy of 74.26% and an agreement rate of 74.78% with *MainNet*, whereas under the same feature dimensionality the ANN *GuardNet* reaches 76.84% accuracy and 77.13% agreement. In the current software implementation, the SNN-based *GuardNet* is also slower—the measured per-sample latency is 6.97 ms versus 0.50 ms for the ANN *GuardNet*, although the two checkpoints are comparable in size (338.8 KB vs. 355.9 KB). Figure 16 summarizes this preliminary comparison.

These results suggest a balanced reading. On the positive side, they indicate that the *GuardNet* learning framework of *LipCache* is compatible with more than one model family: the guard does not necessarily have to remain an ANN. At the same time, the current SNN study should be understood as exploratory evidence rather than a deployment-level conclusion—the SNN-based *GuardNet* has not yet reached the predictive quality required for reliable cache hits, and its measured latency was obtained on a conventional PyTorch/SpikingJelly software stack rather than

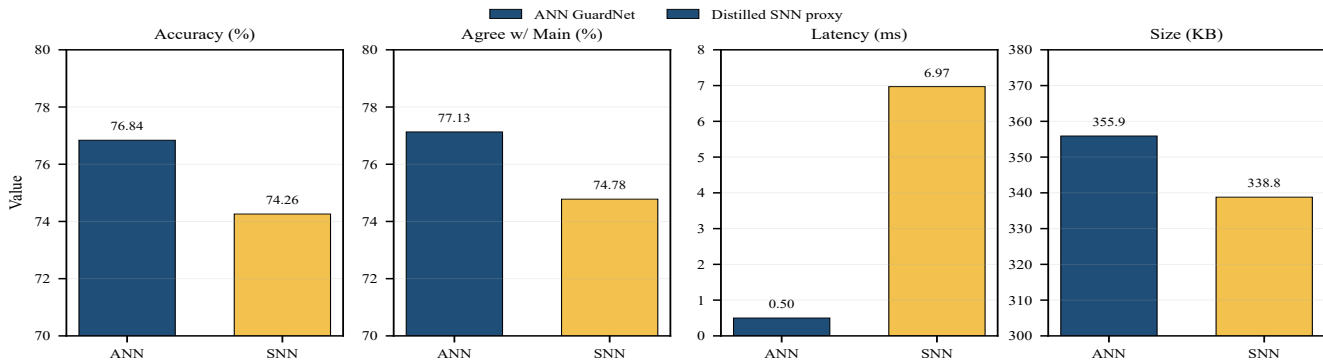


Fig. 16. Preliminary comparison on CIFAR-10 between the ANN-based *GuardNet* and the currently distilled SNN-based *GuardNet*: (a) classification accuracy, (b) consistency with *MainNet*, (c) per-sample latency, (d) checkpoint size.

on event-driven hardware. The current evidence therefore supports feasibility, not superiority.

Based on these preliminary observations, we outline below a number of open questions in future work.

First, energy-efficiency evaluation requires a unified and physically meaningful accounting rule. Directly comparing a heuristic spike-count estimate of an SNN with a FLOP-style approximation of an ANN is insufficient. Both models should be analyzed under an operation-level formulation—ANN inference is measured in multiply-accumulate operations, and event-driven SNN inference is measured in accumulate operations [38], [39].

Second, the SNN-based *GuardNet* itself needs significant improvement before it can be integrated as a reliable guard. Promising directions include stronger distillation objectives, better temporal-encoding schemes, explicit margin-aware training tailored to spiking features, and meaningful architectural constraints capable of recovering the Lipschitz control used by the ANN *GuardNet*.

Third, the resulting energy and performance claims should be validated on neuromorphic hardware, including Loihi, TrueNorth, or SpiNNaker-class systems [40], [41], [42]. Such studies should go beyond coarse spike statistics and explicitly account for synaptic events, routing overhead, memory access, and board-level power measurement.

Finally, an improved SNN-based *GuardNet* should be integrated into the complete *LipCache* inference pipeline, so that its impact on hit rate, fallback rate, end-to-end latency, and overall throughput can be evaluated at the full-system level rather than only at the standalone level.

Taken together, these directions point to a broader research path: *LipCache* evolving from a cache-assisted acceleration framework into an energy-aware edge inference system. If *GuardNet* modeling, cache design, and hardware deployment can be jointly optimized, *LipCache* may ultimately become a unified solution that balances accuracy, latency, throughput, and energy.

8 ACKNOWLEDGMENT

Z. Xiang conceived the *LipCache* idea and developed the theoretical proofs. Y. Chen and F. Ying conducted the experiments, produced the figures, H. Zhao contributed to the

framework design. Z. Xiang and Y. Chen proposed the SNN-based *GuardNet*, and Y. Chen carried out the preliminary experiments. B. Zhou and S. Dustdar played an important role in problem modeling and feasibility validation. The manuscript was polished with the assistance of generative-AI tools (GLM 5.2¹ and DeepSeek V4²).

REFERENCES

- [1] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, no. 4, pp. 216–222, 2018.
- [2] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: A review," *Proceedings of the IEEE*, vol. 111, no. 1, pp. 42–91, 2023.
- [3] H.-I. Liu, M. Galindo, H. Xie, L.-K. Wong, H.-H. Shuai, Y.-H. Li, and W.-H. Cheng, "Lightweight deep learning for resource-constrained environments: A survey," *ACM Computing Surveys*, vol. 56, no. 10, pp. 1–42, 2024.
- [4] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang, and W. Jia, "Empowering edge intelligence: A comprehensive survey on on-device ai models," *ACM Computing Surveys*, vol. 57, no. 9, pp. 1–39, 2025.
- [5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
- [6] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.
- [7] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464–2469.
- [8] S. S. Ogden, G. Gilman, R. J. Walls, and T. Guo, "Many models at the edge: Scaling deep inference via model-level caching," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2021, Washington, DC, USA, September 27 - Oct. 1, 2021*, E. El-Araby, V. Kalogeraki, D. Pianini, F. Lassabe, B. Porter, S. Ghahremani, I. Nunes, M. Bakhouya, and S. Tomforde, Eds. IEEE, 2021, pp. 51–60.
- [9] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "No-Scope: Optimizing deep CNN-based queries over video streams at scale," in *Proceedings of the VLDB Endowment*, vol. 10, no. 11, 2017, pp. 1586–1597.
- [10] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu, "Deepcache: Principled cache for mobile deep vision," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 129–144.

1. <https://www.bigmodel.cn>

2. <https://chat.deepseek.cn>

- [11] Y. Li, C. Zhang, S. Han, L. L. Zhang, B. Yin, Y. Liu, and M. Xu, "Boosting mobile CNN inference through semantic memory," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 2362–2371.
- [12] M. Parger, C. Tang, C. D. Twigg, C. Keskin, R. Wang, and M. Steinberger, "Deltacnn: End-to-end cnn inference of sparse frame differences in videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 497–12 506.
- [13] H. Sedghi, V. Gupta, and P. M. Long, "The singular values of convolutional layers," in *International Conference on Learning Representations*, 2019.
- [14] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations*, 2018.
- [15] K. Sun, X. Wang, X. Miao, and Q. Zhao, "A review of AI edge devices and lightweight CNN and LLM deployment," *Neurocomputing*, vol. 614, p. 128791, 2025.
- [16] X. Zhang, R. Razavi-Far, H. Isah, A. David, G. Higgins, and M. Zhang, "A survey on deep learning in edge-cloud collaboration: Model partitioning, privacy preservation, and prospects," *Knowledge-Based Systems*, vol. 310, p. 112965, 2025.
- [17] J. Fang, Y. An, Y. Liu, Z. Teng, X. Zhai, H. Tang, and H. Chen, "MAE: Collaborative inference acceleration with efficient DNN partitioning and resource allocation in resource-constrained edge computing," *Computer Networks*, vol. 278, p. 112073, 2026.
- [18] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.
- [19] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *International Conference on Learning Representations*, 2018.
- [20] M. Hassan, S. Davy, M. Zawish, O. B. Zuber, and N. Ashraf, "Neucodex: Edge-cloud co-inference with spike-driven compression and dynamic early-exit," in *International Conference on Machine Learning and Applications, ICMLA 2025, Boca Raton, FL, USA, December 3-5, 2025*. IEEE, 2025, pp. 784–789.
- [21] S. Venugopal, M. Gazzetti, Y. Gkoufas, and K. Katrinis, "Shadow puppets: Cloud-level accurate AI inference at the speed and economy of edge," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. Boston, MA: USENIX Association, 2018.
- [22] Z. Huang, F. Dong, X. Guo, and D. Yin, "Fasei: Fast serverless edge inference with synergistic lazy loading and layer-wise caching," in *44th IEEE International Conference on Computer Communications, INFOCOM 2025, London, United Kingdom, May 19-22, 2025*. IEEE, 2025.
- [23] A. K. Sinthia, N. I. Mahbub, M. N. Sultan, and E. Huh, "Cachemoe: Task-aware expert model caching for multitask inference in distributed edge iot networks," *IEEE Internet Things J.*, vol. 12, no. 24, pp. 55 725–55 741, 2025.
- [24] Q. Chen, X. Chen, and K. Huang, "Slimcaching: Edge caching of mixture-of-experts for distributed inference," *IEEE Trans. Mob. Comput.*, vol. 25, no. 7, pp. 10 924–10 938, 2026.
- [25] Y. Wu, H. Zhang, and H. Huang, "RetrievalGuard: Provably robust 1-nearest neighbor image retrieval," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 24 266–24 279.
- [26] A. Araujo, A. J. Havens, B. Delattre, A. Allauzen, and B. Hu, "A unified algebraic perspective on lipschitz neural networks," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [27] H. Zhang, T. Weng, P. Chen, C. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems 31*, 2018.
- [28] J. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, 2019, pp. 1310–1320.
- [29] Y. Huang, H. Zhang, Y. Shi, J. Z. Kolter, and A. Anandkumar, "Training certifiably robust neural networks with efficient local lipschitz bounds," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 22 745–22 757.
- [30] M. Fazlyab, T. Entesari, A. Roy, and R. Chellappa, "Certified robustness via dynamic margin maximization and improved lipschitz regularization," in *Advances in Neural Information Processing Systems*, 2023.
- [31] K. Hu, K. Leino, Z. Wang, and M. Fredrikson, "A recipe for improved certifiable robustness," in *The Twelfth International Conference on Learning Representations*, 2024.
- [32] Z. Wang, B. Hu, A. J. Havens, A. Araujo, Y. Zheng, Y. Chen, and S. Jha, "On the scalability and memory efficiency of semidefinite programs for lipschitz constant estimation of neural networks," in *The Twelfth International Conference on Learning Representations*, 2024.
- [33] Y. Xu and S. Sivaranjani, "Eclipse: Efficient compositional lipschitz constant estimation for deep neural networks," in *Advances in Neural Information Processing Systems*, vol. 37, 2024.
- [34] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.
- [35] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," CS231N Course Project, Stanford University, 2015.
- [36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
- [37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [38] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in Neuroscience*, vol. 14, p. 119, 2020.
- [39] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.
- [40] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [41] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.
- [42] T. Matsuo, "Neuromorphic processors for SNN-based edge AI: A review," *Micromachines*, vol. 15, no. 1, p. 21, 2024.



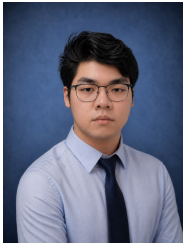
Zhengzhe Xiang received the B.S. and Ph.D. degrees in computer science and technology from Zhejiang University, Hangzhou, China. He was a Visiting Scholar with Shanghai Jiao Tong University, Shanghai, China, in 2022. He is currently an Associate Professor with Hangzhou City University, Hangzhou, China. His research interests focus on service computing and edge computing. He serves as a reviewer for several international journals, such as IEEE Transactions on Mobile Computing, IEEE Transactions

on Services Computing, IET Communications, and Digital Communications and Networks. He also serves as a program committee member for many international conferences.



Schahram Dustdar is a Full Professor of Computer Science and heads the Distributed Systems Group at TU Wien, Vienna, Austria. He is an ICREA Research Professor at UPF Barcelona, Spain. His research interests include distributed systems, edge intelligence, and complex and autonomous software systems. He is the Editor-in-Chief of Computing; an Associate Editor of ACM Transactions on the Web, ACM Transactions on Internet Technology, IEEE Transactions on Cloud Computing, and IEEE

Transactions on Services Computing. He also serves on the editorial boards of IEEE Internet Computing and IEEE Computer. He received the ACM Distinguished Scientist Award, the Distinguished Speaker Award, and the IBM Faculty Award. He is an elected member of Academia Europaea and served as the Chair of its Informatics Section from 2015 to 2022. He is an IEEE Fellow and an AAIA Fellow, and currently serves as the President of AAIA.



Yinlin Chen is currently pursuing the M.S. degree with the School of Computer and Computational Sciences, Hangzhou City University, Hangzhou, China. His research interests include edge computing, large language model applications, and distributed intelligent systems, with a focus on application optimization, task offloading, and intelligent resource scheduling for large language models in resource-constrained edge computing environments.



Fuli Ying is currently pursuing the M.S. degree with the School of Computer and Computational Sciences, Hangzhou City University, Hangzhou, China. Her research interests span mobile edge computing, deep-learning architectures, and image processing, with a focus on developing efficient algorithms for distributed computing environments and intelligent visual analytics systems.



Binbin Zhou received the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2021. She is currently an Associate Professor with the School of Computer Science and Computing, Hangzhou City University, Hangzhou, China. Her main research areas include spatiotemporal deep learning, multimodal fusion, artificial intelligence, and brain-inspired computing.



Hailiang Zhao (Member, IEEE) received the PhD degree in Computer Science and Technology in 2024. He is currently an assistant professor with the School of Software Technology, Zhejiang University, China. His research interests include distributed computing and services computing. He has published several papers in flagship conferences and journals such as IEEE ICWS, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Mobile Computing, IEEE Transactions on Services

Computing, Proc. IEEE, etc. He was a recipient of the Best Student Paper Award of IEEE ICWS 2019.