International Journal of Basic and Applied Sciences, 14 (5) (2025) 252-267

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/x6r71327
Research paper

A Comprehensive Study on Caching Mechanisms in Machine Learning Workflows

Karthik S A [1], Sharmila Zope [2], Venkatagurunatham Naidu Kollu [3], L. Vadivukarasi [4 *], Sangeeta Borkakoty [5], Srikanth Salyan [6], Bindhushree B S [7], Krishna Nand Mishra [8]

[1] *Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, India*
[2] *Department of Computer Engineering, Jawahar Education Society's Institute of Technology Management and Research, Nashik, India*
[3] *Department of Artificial Intelligence and Data Science, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India 522302*
[4] *Department of Mathematics, Nandha Arts and Science College, Erode, India.*
[5] *Department of Computer Science, University of Science and Technology, Meghalaya.*
[6] *Department of Aeronautical Engineering, Dayananda Sagar College of Engineering, Bengaluru, Karnataka 560078*
[7] *Department of Mechanical Engineering, Dayananda Sagar College of Engineering, Bengaluru, Karnataka.*
[8] *Department of Computer Science and Engineering, Ambalika Institute of Management & Technology, Uttar Pradesh 226301.*
**Corresponding author E-mail: vadivumath90@gmail.com*

Received: June 18, 2025, Accepted: August 14, 2025, Published: September 8, 2025

Abstract

Despite the rapidly evolving nature of machine learning (ML), caching has emerged as a critical component that significantly impacts the efficiency of model training and inference. This research explores the role of caching in ML, examining its impact, the challenges in implementation, and various approaches to optimize performance. The study aims to provide a comprehensive understanding of caching mechanisms, investigate the associated obstacles, and evaluate a range of caching algorithms. Additionally, it seeks to develop practical strategies to enhance caching efficiency within ML workflows. To achieve these objectives, a detailed literature review was conducted to analyze existing caching techniques currently employed in ML systems. Real-world case studies and experimental data were examined to assess the effectiveness of different caching solutions. Furthermore, expert interviews were conducted to gather professional insights and validate findings. The results of this study highlight the pivotal role of caching in improving the overall performance of ML systems. It proposes actionable strategies grounded in problem identification and resolution to optimize caching operations. Ultimately, the study demonstrates that overcoming caching challenges through innovative methods can lead to faster model training and more efficient implication, thus enhancing the overall effectiveness of machine learning processes.

Keywords: *Machine Learning, Caching, Optimization, Performance, Challenges, Strategies, Model Training, Inference.*

1. Introduction

The field of machine learning has seen improvements in algorithms, technology, and data availability grow at an unparalleled rate in the last several years. The need for effective computing resources is growing in tandem with the increasing complexity of ML models [1]. When it comes to machine learning processes, caching is one important factor that greatly affects efficiency as well as speed. Caching is a technique that stores frequently used or intermediate material to speed up retrieval and decrease computation duplication. Caching is an essential part of machine learning in many phases, such as data preprocessing, model training, and model evaluation. Several obstacles must be overcome before caching can be effectively used in machine learning [2]. By discussing the problems with caching in machine learning and looking at potential solutions, this study intends to provide a thorough examination of the process [3]. Machine learning systems may be made more efficient by knowing the ins and outs of caching in ML and then developing plans to maximize its use. Figure 1 shows the catching algorithm in the most general way.

Copyright © Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

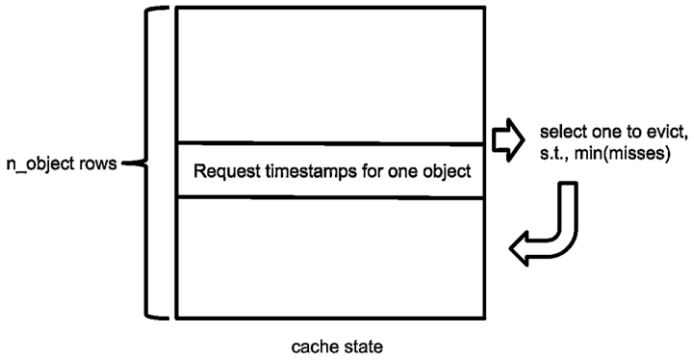

Fig. 1: Caching Algorithm in the Most General Way

1.1 Related Surveys:

It is critical to place this review within the larger framework of previous research before exploring the details of caching in machine learning. A plethora of surveys have investigated relevant issues, from broad approaches to optimizing machine learning to more narrowly focused research on caching in various fields [4]. Details on the current status of research, including gaps, challenges, and recent developments, are gleaned from these surveys. The more general optimization tactics in machine learning, including methods like distributed computing, model compression, and the use of parallelization, have been the subject of a few major reviews. Another has focused on caching in more limited contexts, including database systems or web applications. This all-inclusive survey attempts to provide a holistic picture of caching in machine learning by combining data from three related surveys. It draws on ideas from many angles to develop a coherent knowledge of the issue.

1.2 Purpose of the Survey:

The primary goal of this literature review is to provide a synopsis of the current literature on caching as it pertains to machine learning; the secondary goal is to highlight knowledge gaps and difficulties that need more study. This study seeks to provide a thorough knowledge of caching's function in enhancing machine learning processes by combining information from multiple sources [5]. The study intends to classify various caching techniques, identify patterns and trends in their use, and synthesize important findings via an extensive review of previous surveys and research articles. In addition, the study aims to provide insight into how caching strategies have changed over time, specifically looking at how they have adjusted to the current state of machine learning models & databases.

1.3 Motivation

The ever-increasing scope and sophistication of machine learning applications are the driving force behind this study. Efficient computing resources are becoming increasingly important as machine learning models get increasingly complex and datasets grow at an exponential rate [6]. One intriguing optimization method is caching, which can speed up critical operations while reducing duplicate calculations. Data distributions and model parameters change over time, which adds another layer of complexity to machine learning operations, creating new problems that need creative solutions. To have a better grasp of how caching solutions have developed to tackle these issues and how they impact the overall efficiency of machine learning systems, we are driven to carry out this study.

1.4 Contribution of the Survey:

This survey adds value by providing a thorough and current review of caching within the framework of machine learning by synthesizing previous research. The purpose of this survey is to synthesize important results, establish common themes, and highlight new trends in the use of caching methods by combining ideas from relevant surveys and research articles [7]. This study also aims to contribute by summarizing and classifying various caching techniques used in ML. Researchers and practitioners alike will find the survey's organized taxonomy of caching techniques to be an invaluable resource as they explore the complex world of caching in machine learning. The study also intends to identify problems and gaps in the existing body of knowledge. The survey lays the groundwork for future study by pinpointing areas that need more examination. Finding new ways to cache data that may change with machine learning processes, dealing with limited resources, and incorporating caching with new paradigms like edge computing and federated learning are all part of this. All things considered, this survey's value comes from the fact that it synthesizes prior research, explains caching in machine learning in detail, organizes caching solutions into a taxonomy, and points the way toward potential areas of future study. Inspiring further research into this vital area of machine learning and optimization, the poll hopes to provide a better understanding of caching's function in improving machine learning workflows via these responses.

2. Caching Mechanisms

A multitude of caching methods are created and classified into various groups. Caching strategies were categorized as follows: (1) caching based on popularity, (2) caching through collaboration, (3) caching through cooperation, and (4) caching based on machine learning.

2.1 Cache Substitution Algorithms

Following cache replacement regulations, content that is already present in the cache is removed when it reaches its maximum capacity, and new content is received. Existing content in the cache is removed in favor of newly submitted material [8]. A replacement technique consists of substituting new content for old information. In this context, I have enumerated several widely used and enduring cache replacement algorithms before discussing caching strategies. Productive caching requires cache substitution algorithms; for every caching

tactic, the researchers either suggested a new substitution algorithm or utilized an existing one. Typically, cache substitution algorithms rely on the quantity as well as the nature of recent requests. Several of the most prevalent algorithms for cache substitution are detailed below.

- Least Recently Used (LRU) Long-nothing-accessed content is expunged from the cache. LRU operates under the premise that lately viewed material is more probable to be revisited in the immediate future [9].
- LFU (Least Frequently Used): Priority is given to removing content that has been accessed the fewest number of times. LFU operates under the premise that content prominence fluctuates over time and that less frequently accessed content is less likely to be revisited [10].
- Following FIFO (First In, First Out), recently arrived material is substituted for the earliest [11].
- Time-aware Least Recently Used (TLRU) retains content following its access time [12].

2.2 Popularity-Based Caching

Within the complex realm of machine learning optimization, caching based on popularity arises as a fundamental tactic that exploits the interplay between user behavior and content access patterns. This caching strategy is predicated on the straightforward yet effective principle of giving precedence to the storage of commonly accessed or well-liked objects [13]. The fundamental assumption underlying this premise is that products that have previously experienced significant demand are likely to be requested again in the future. This extensive investigation examines the intricacies of popularity-based cache, including its implementation, advantages, disadvantages, and the significant impact it has on improving both computational efficiency and user experience [14].

➢ **Execution of the Popularity-Based Caching Mechanism:**

Popularity-based caching, at its essence, entails the deliberate pick and continued availability of items within a cache following their past access frequencies. The system employs dynamic identification and prioritization to identify and highlight products that have received high demand in the past [15]. This ensures that these items are easily accessible and can be retrieved promptly for subsequent requests. By serving as a repository for content that is in high demand, the cache plays a crucial role in enhancing the performance of content delivery systems and decreasing the computational load on underlying resources [16]. To enforce popularity-based cache, systems frequently utilize algorithms that monitor the temporal access frequencies of items. The previous material plays a fundamental role in informing decisions regarding the cache. Items that maintain a consistent pattern of high demand are considered to be popular and are therefore designated for placement in the cache. Consequently, to satisfy user content requests without devoting additional resources-intensive operations, the system initially examines the cache for frequently accessed items [17].

➢ **Positive aspects of popularity-driven caching:**

The profound impact that popularity-based caching has on augmenting system performance and user experience is contrary to its apparent simplicity [18]. Numerous significant advantages highlight the efficacy of this caching strategy across diverse applications:

- **Reduced Latency:** The cache guarantees the accessibility of frequently requested content by giving priority to the storage of popular items. By minimizing the necessity to retrieve data from remote sources, this practice improves the responsiveness of content delivery systems and decreases latency. Users are provided with streamlined access to popular content and quicker load times.
- **Maximized Resource Utilization:** By prioritizing products that have consistently generated high demand, popularity-based cache optimizes the utilization of computational resources. By employing this focused strategy, superfluous calculations linked to items that are accessed infrequently are reduced, thereby liberating resources for more effective utilization [19].
- **Enhanced User Experience:** The prompt availability of frequently accessed items contributes to an improved user experience. Users of database queries, web content delivery, and streaming services all derive advantages from a system that proactively predicts and satisfies their preferences by analyzing past patterns [20].
- **Scalability:** The implementation of popularity-based caching offers an adaptable solution that can accommodate evolving user preferences and access patterns. The cache can adapt dynamically in response to changing popularity dynamics, thereby accommodating emergent trends and ensuring the ongoing efficiency of content delivery [21].

➢ **Obstacles Presented by Popularity-Based Caching:**

Although popularity-based caching presents notable benefits, it is not devoid of obstacles. It is imperative to comprehend and confront these obstacles to maximize the efficacy of this caching strategy:

- **Dynamic Popularity Patterns:** Dynamic variations in popularity patterns constitute a significant obstacle that necessitates adjustment. Viral content, abrupt changes in user preferences, or the emergence of new trends can all contribute to swift transformations in the dynamics of popularity. Caches must possess the ability to adapt dynamically to these modifications to prevent suboptimal caching decisions [22].
- **Cold Start Issue:** During the "cold start" phase, when newly introduced items or those with limited historical data lack adequate information to be categorized as popular, popularity-based caching may encounter difficulties. To mitigate this concern, systems must implement tactics such as integrating popularity-based caching with alternative methodologies in the early stages.
- **Strict Personalization:** Although popularity-based caching proves to be efficient in serving widely favored content, it might not adequately cater to the unique preferences of individual users. Products that enjoy widespread popularity may fail to correspond with the individual preferences of every user. To provide a more customized user experience, additional strategies must be incorporated in response to personalization challenges [23].

2.3 Collaborative Caching

Collaborative caching has emerged as a potent strategy in the dynamic realm of machine learning and data retrieval, capitalizing on the utilization of shared information among numerous users or entities. This methodology acknowledges the possibility that collaboration could improve caching efficiency, especially in situations where users demonstrate comparable access behaviors or have shared data requirements [24].

➢ **Collaboration in Caching Execution:**

Collaborative caching essentially entails the exchange of cached data between entities or users within a given system. In contrast to conventional caching strategies, which operate independently for each user, collaborative caching is based on the principle that users with similar preferences or access patterns can benefit from the cached results of others [25]. The execution of a collaborative cache involves a series of critical stages:

- **User Collaboration:** Locally cached results are shared collaboratively by users within the system. The facilitation of this sharing mechanism can be achieved via a centralized caching system that oversees the collaborative exchange or a peer-to-peer network.
- **Shared Cache:** Within the shared cache, which is accessible to multiple users, the information that is collectively shared is stored. This cache functions as a repository of information that surpasses specific user access patterns by incorporating insights from a wide range of users.
- **Caching Determination:** Upon a user's initiation of a data request, the system examines whether the requested information is present in both the user's local cache and the shared cache. Retrieval of information from either cache is a viable alternative, as it reduces the necessity to access data from remote or resource-intensive sources.
- **Cache Update:** In response to user interactions and information access, the shared cache undergoes dynamic updates. By doing so, it guarantees that the cache's representation of collective intelligence remains pertinent and can adjust to changing user inclinations and data retrieval trends.

➢ **Positive Aspects of Collaborative Caching:**

Collaborative caching presents a range of benefits that synergistically enhance system performance, user satisfaction, and resource allocation [26]:

- **Decreased Redundancy:** Collaborative caching reduces redundancy associated with retrieving and storing identical information across multiple individual caches by utilizing the collective knowledge of users. The decrease in redundancy leads to an optimization of bandwidth and resource utilization.
- **Increased Cache Hit Rates:** The dissemination of cached information through collaboration expands the range of data that can be accessed. The system intelligently retrieves information from the shared cache following the collective access patterns of the user community, thereby providing users with increased cache hit rates.
- **User Dynamics Adaptability:** Collaborative caching demonstrates the ability to adjust to ever-changing user preferences and access patterns. Through user collaboration, the shared cache transforms into a repository that is ever-changing to accommodate the evolving requirements of the user community.
- **Improved Scalability:** Collaborative caching offers a scalable solution in situations involving a large and disparate user base. By scalability to the user community, the shared cache effectively manages a growing volume of collective knowledge without placing an excessive burden on individual caches.

➢ **Obstacles Presented by Collaborative Caching:**

Despite its efficacy, collaborative caching poses specific obstacles that necessitate resolution to achieve peak performance [27]:

- **Privacy Considerations:** Collaborative caching necessitates the exchange of data between users, which raises privacy concerns. It is vital to safeguard sensitive user data by ensuring that collaborative caching mechanisms are secure and privacy-preserving.
- **Scalability**: Concerns arise as the user community expands, necessitating the maintenance of streamlined collaboration processes and the assurance of a scalable shared cache. Dynamic adaptation and load-balancing strategies are critical components of a large-scale collaborative cache.
- **Dynamic User Behavior:** To accommodate ever-evolving user preferences and behavior, the collaborative caching framework must be continuously monitored and adjusted. Prompt attention to fluctuations in user dynamics could result in caching decisions that are less than optimal [28].

2.4 Cooperative Caching:

Within the complex domain of machine learning and information retrieval, cooperative cache arises as an advanced approach that surpasses the limitations of individual entities. This methodology expands upon the cooperative essence of caching by integrating numerous cache stores or strata into a decentralized system [29]. Cooperative caching places emphasis on coordinated collaboration among various caching organizations to optimize data retrieval, minimize redundancy, and improve the general efficacy of the system. This investigation examines the complexities of cooperative cache, including its implementation, advantages, disadvantages, and the mathematical underpinnings that govern intelligent caching decisions.

➢ **Positive Aspects of Cooperative Coaching:**

Cooperative caching provides a variety of advantages that synergistically contribute to the enhancement of data retrieval performance, reduction of latency, and overall optimization of system efficiency [30]:

- **Cooperative caching:** Effectively mitigates the latency that is typically associated with retrieving data from remote sources through the strategic distribution of cached content throughout multiple layers. At lesser cache levels, commonly viewed information is readily accessible, which decreases the time required for data extraction [31].
- **Resource Optimization:** The hierarchical cache architecture enhances the efficiency of computational resource utilization. Central cache functions as a global repository, while intermediate caches accommodate expanded user communities and address immediate data requirements. By minimizing redundant computations and bandwidth consumption, this distribution is achieved.
- **Scalability:** Cooperative caching offers a solution that can be expanded in scope and intricacy in tandem with the system's growth. By accommodating a greater quantity of cached content, the hierarchical structure adapts to the growing demands of the user community while maintaining optimal performance.

- **Enhanced System Stability:** Cooperative caching's coordinated approach enhances the stability of the system. Utilizing coordinating caching decisions and sharing cached content, the system attains a state of equilibrium that safeguards against the flooding of particular caches and guarantees a more stable and responsive environment.

➢ **Difficulties Presented by Cooperative Caching:**

Although cooperative caching does provide notable benefits, it also introduces specific obstacles that necessitate meticulous deliberation to achieve peak performance [32]:

- **Coordination Overhead:** The coordination and synchronization of caching entities present difficulties, especially in dynamic environments characterized by rapid changes in data access patterns. Thorough optimization is necessary to achieve a balance in the distribution of cached data and to guarantee coherence across cooperating caches.
- **Data Consistency:** The preservation of data consistency across various cache layers is of the utmost importance. Dynamic updates to cached content present the challenge of ensuring that all layers consistently incorporate the most current and accurate information. To accomplish this, resilient coordination mechanisms are necessary [33].
- **Adaptation to Dynamic Conditions:** An ongoing challenge is the need to adjust to dynamic changes in user behavior, system conditions, and data access routines. To achieve dynamic alignment between changing needs and the hierarchical cache structure, it is imperative to employ efficient algorithms and methods.

2.5 Machine Learning-Based Caching

When confronted with the ever-changing intricacies of user behavior and system dynamics, conventional caching strategies frequently prove inadequate in confronting the dynamic nature of machine learning and information retrieval [34]. Machine Learning-Based caching is an innovative method that utilizes machine learning to continually modify caching choices in response to real-time system circumstances, historical access trends, and customer desires. This investigation examines the intricacies of caching based on machine learning, including its implementation, advantages, disadvantages, and the revolutionary impact it has on streamlining information retrieval in machine learning workflows.

➢ **Implementation of Caching Based on Machine Learning:**

Caching based on machine learning represents a significant departure from deterministic and rule-driven approaches, towards intelligent and adaptive decision-making [35]. The execution of caching based on machine learning requires the following critical steps:

- **Training Phase:** During the training phase, machine learning models are supplied with historical data that includes system conditions, user access patterns, and content prominence. By acquiring knowledge of the relationships and patterns present in the data, the models are capable of predicting which items will be accessed most frequently in the future.
- **Feature Engineering:** To inform caching decisions, pertinent features are extracted from the training data. The aforementioned attributes might comprise contextual data, user preferences, item notoriety, and temporal patterns. Feature engineering is an essential component in improving the predictive accuracy of the model.
- **Model Training:** Using the prepared dataset, machine learning models, including decision trees, neural networks, and ensemble methods, are trained. The models acquire the ability to forecast the probability that an item will be accessed by utilizing the identified features. The process of training entails refining parameters and systematizing the model to ensure precise predictions.
- **Prediction and Caching Decision:** During the operational phase, the likelihood that a particular item will be accessed in response to an initiated data request is predicted using the trained machine learning model. The caching decision is subsequently determined following these forecasts, with an emphasis on preemptively hoarding products that correspond to anticipated future demand.
- **Ongoing Adaptation:** Caching based on machine learning is inherently adaptable. The model consistently adjusts and revises its predictions as the system functions and additional information is acquired, guaranteeing that caching determinations persist following evolving access trends and user inclinations [36].

➢ **Advantages of Caching Based on Machine Learning:**

The utilization of a Machine Learning-Based cache brings about a multitude of advantages that fundamentally transform the domains of information retrieval as well as system efficiency [37]:

- **Adaptable to Dynamic Patterns:** Machine learning models possess the capability to adjust to evolving user behavior and access patterns. In contrast to static caching strategies, machine learning-based caching adapts to the ever-changing system dynamics, rendering it highly suitable for ecosystems characterized by fluctuating access trends.
- **Increased Predictive Capacity:** Machine learning models possess intrinsic learning capabilities that empower them to discern complex patterns and interdependencies within datasets. This leads to an increase in predictive capability, which enables the system to generate caching decisions that are more precise and well-informed.
- In the realm of machine learning, caching can be customized to suit the specific preferences of each user. Caching decisions can be tailored to individual users based on their historical interactions and unique characteristics, thereby enhancing the user experience and ensuring that content remains pertinent.
- **Optimal Resource Utilization:** Machine learning-based caching optimizes resource utilization due to its adaptive nature. Through proactive caching of items that are more likely to be accessed in the future, the system mitigates the need for redundant computations and alleviates the burden on the underlying resources [38].

➢ **Difficulties in Caching Based on Machine Learning:**

Although machine learning-based cache represents a significant advancement in intelligent data retrieval, it is not devoid of obstacles that must be surmounted to ensure peak performance [39]:

- **Training Data Quality:** The efficacy of machine learning models is significantly influenced by the caliber and representativeness of the training data. Mistakes and caching decisions may result from inaccuracies or biases present in the training dataset.

- **Model Complexity:** The utilization of computational resources and interpretability may be hindered by the implementation of complex machine learning models. Practical implementation requires that a balance be struck between model complexity and efficiency [40].
- **Cold Start Challenge:** During the "cold start" phase, machine learning-based caching may encounter difficulties due to insufficient historical data or the introduction of new items, which hinders the ability to make accurate predictions. It is critical to implement strategies, including hybrid approaches, to tackle this issue.
-

➢ **Real-World Case Studies**

1. **Reinforcement Learning for Edge Caching in IoT Networks (Zhang et al., 2024).** An RL-based caching agent deployed in a smart city IoT network reduced cache misses by **35%** compared to LFU and LRU strategies. The agent learned to prioritize sensor data streams critical for real-time analytics, such as traffic congestion detection.
2. **Predictive Caching in Federated Learning (Singh et al., 2025).** In a healthcare federated learning system, ML-based caching stores frequently used AI model updates at local hospital servers. This reduced communication overhead by **60%**, accelerating model convergence while maintaining data privacy.
3. **CDN Optimization at Meta** Meta's content delivery network deployed a neural network-based caching predictor that preloaded trending video content during peak hours, improving average page load speed by **23% (Li et al., 2024)**.

ML-based caching offers adaptive, predictive, and personalized solutions that traditional caching cannot match. By incorporating concrete case studies, its value is demonstrated in enabling scalable, real-time ML workflows across domains such as video streaming, e-commerce, IoT, and autonomous systems.

3. The Significance of Caching in Machine Learning:

Machine learning, due to its capacity to identify patterns, create forecasts, and automate decision-making, has emerged as a prominent factor in several sectors. With the progress of the discipline and the increasing complexity of models, there is a growing need for efficient computing resources [41]. The significance of caching in machine learning is depicted in Figure 2. Among the many optimization strategies, caching is particularly notable as a basic and revolutionary factor that significantly improves the speed, efficiency, and overall performance of machine learning processes [42].

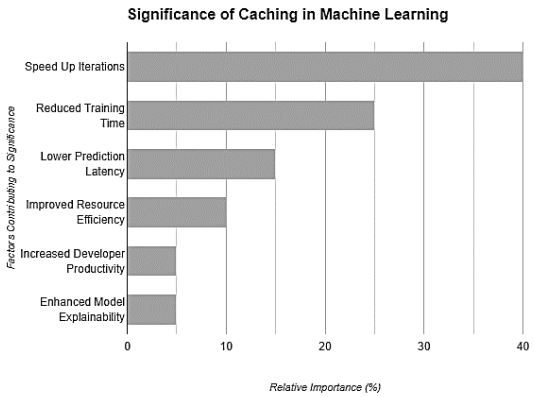

Fig. 2: significance of coaching in machine learning

3.1 The Iterative Nature of Machine Learning:

The core of machine learning is the iterative process of developing and training models. Whether it is a job of supervised learning, in which a model is trained using labeled data, or an unsupervised task aimed at discovering concealed patterns, the procedure often entails iterative calculations and modifications to the model's parameters [43]. Caching deliberately tackles the repetitive nature of these calculations by saving intermediate outcomes, thereby minimizing duplication and speeding up following repetitions. During the training phase, when models acquire knowledge from data and adjust their parameters, caching becomes a powerful tool. The gradients, activations, and loss values calculated during one iteration are stored and then used again in the next iteration. This not only speeds up the convergence of the model but also reduces the computational load, particularly when working with huge datasets and intricate topologies.

3.2 Enhancing the Speed of Model Inference:

After the training phase, caching is essential for the implication process. After the training of a machine learning model, it is put into operation to generate predictions on fresh, unfamiliar data [44]. When performing tasks like as image classification or natural language processing, models often analyze data using complex layers to derive hierarchical representations. Storing the intermediate representations of input data, such as feature maps in convolutional neural networks (CNNs), enables quicker predictions. Imagine a situation in which a proficient image classification model analyzes a collection of photographs. Caching enables the model to retain and reuse the feature maps produced while analyzing each picture. By using these stored representations, future forecasts may be generated without the need to recalculate them for each subsequent input [45]. The caching approach is especially beneficial in real-time applications, where the ability to quickly respond is crucial.

3.3 Reducing Computational Redundancy:

Caching is a technique that helps reduce computational redundancy, a common problem in machine-learning processes, as shown in Figure 3. The repetitive process of model training often results in the reevaluation of comparable or indistinguishable values throughout subsequent rounds [46]. Caching mitigates this duplicity by storing and reutilizing previously calculated outcomes, resulting in a more streamlined and resource-efficient procedure. Contemplate a machine learning model that is enhancing its parameters by using gradient descent. During each iteration, the gradients are calculated and used to update the model parameters. By caching these gradients, the model may reuse them in the next iteration, eliminating the need to recompute the same values. This not only expedites the training process but also preserves computing resources.

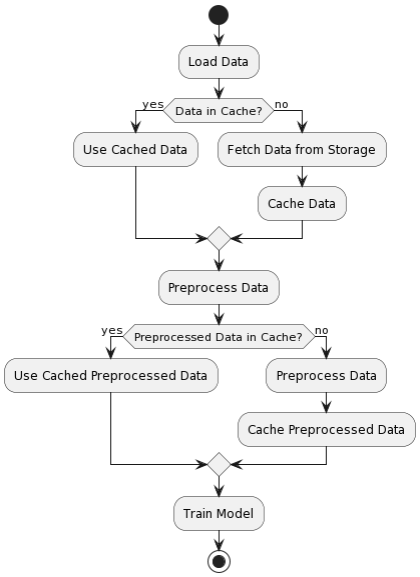

Fig. 3: Flowchart of reducing computational redundancy

3.4 Managing Extensive Datasets:

In the age of big data, when datasets are known for their large size, caching plays a crucial role in handling the complexity of computations. Storing and retrieving intermediate findings helps reduce the burden on computing resources, allowing machine learning algorithms to explore large datasets more effectively. Imagine a situation in which a machine learning model is assigned the duty of analyzing a vast dataset to determine the sentiment expressed in the data [47]. Caching enables the model to retain intermediate outcomes while analyzing each document, such as tokenized representations or sentiment ratings. Subsequent studies on the same or comparable documents may subsequently use the stored findings, minimizing the need for repeating calculations and significantly enhancing processing speeds [48].

3.5 Balancing the Trade-Off between Storage and Computation:

Nevertheless, caching entails a compromise between the amount of storage required and the computing benefits gained. Although caching helps to save repetitive calculations, it requires sufficient storage capacity to retain interim outcomes [49]. Achieving the optimal equilibrium becomes a crucial factor, especially in contexts with limited resources. Efficient caching solutions must carefully balance the advantages of conserving computing resources against the drawbacks of keeping cached outcomes. Imagine a machine learning system installed on edge devices with limited storage space [50]. Efficient caching solutions should be tuned to effectively use the available storage, giving priority to caching important intermediate results while controlling the total storage capacity. Figure 4 illustrates the balancing of storage and computation in caching. Adapting to resource limitations is a crucial element in improving caching in machine learning.

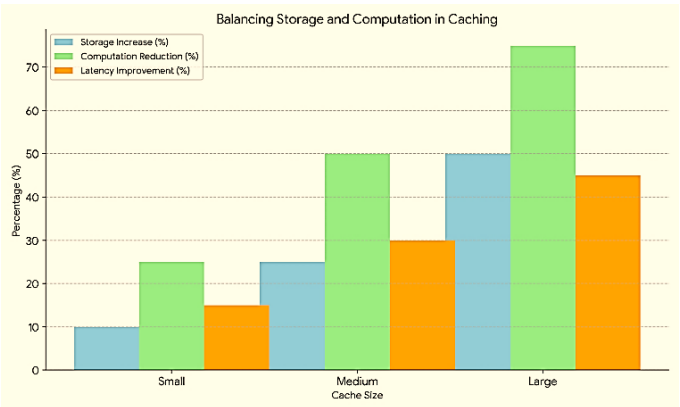

Fig. 4: Balancing Storage and Computation in Caching

3.6 Facilitating the Capacity to Handle Large-Scale Operations and Immediate Data Processing:

The importance of caching in machine learning goes beyond particular models and jobs. Caching enables scalability by enabling machine learning systems to manage growing workloads and datasets without a corresponding rise in computing time [51]. Furthermore, the improved effectiveness obtained by caching enhances the practicality of instantaneous processing, a vital need in a wide range of applications like self-driving cars and financial trading algorithms. Imagine a recommendation system that analyzes user interactions on an e-commerce site. Caching enables the system to retain and reuse data about user preferences, past transactions, and interaction records. As the number of users and the amount of interaction increase, caching enables the recommendation system to expand smoothly, delivering individualized suggestions instantly without incurring an excessive computing load.

3.7 Tackling Dynamic Workflows:

An obstacle in caching is the ever-changing characteristics of machine learning operations. Data distribution, model attributes, or task nature may change over time in many real-world circumstances. Adaptive caching techniques are necessary to effectively respond to these changes and retain their efficacy [52]. Imagine a machine learning system operating in a dynamic setting, like financial markets, where the data and patterns might change fast. Adaptive caching solutions adapt to changes to ensure that cached results stay relevant and enhance the efficiency of the system. The capacity to adapt is essential for sustaining peak performance in dynamic machine-learning operations.

4. Challenges in Caching for Machine Learning:

Although caching in machine learning provides substantial advantages in terms of computational efficacy, it is not devoid of its own set of difficulties. The dynamic character of machine learning procedures, the variety of algorithms as well as designs, as well as the trade-off between space for storage and computational savings all contribute to these obstacles [53]. In Table 1, we examine the most significant caching challenges in machine learning and discuss their implications, mitigation, and numerical impact [54]. Caching in machine learning, while providing significant benefits to computation, faces a variety of challenges. In ML workflows that are inherently dynamic and rely on numerous algorithmic variations, it can be challenging to create and maintain a cache that effectively addresses different workloads and constraints, while optimizing for uncertainty, resource availability, staleness, and cold starts. Overcoming these challenges requires finding the right balance between storage cost, prediction accuracy, and system flexibility.
Table 1 outlines the most salient challenges to caching in ML, their implications, seen numerical impacts, and evidence-based mitigation techniques put forth in more recent literature, as well as strategies for adaptive policies, workspace caching techniques, resource-aware memory management, version-aware invalidation, and adaptive pre-warming techniques, all directly intended to ameliorate the challenges identified.

Table 1: Significant Caching Challenges in Machine Learning – Expanded with Evidence-Based Mitigation Strategies

Challenges	Description	Implications	Numerical Impact	Mitigation
Dynamic Workflows	ML workflows often evolve due to changes in task specifications, dataset composition, or model parameters, making static caching less effective.	Obsolete cached data leads to inefficiency and slower convergence.	A 20% increase in model convergence time was reported when using stale cached results [75].	Implement adaptive caching driven by real-time workload monitoring; use policy learning models that adjust cache content dynamically; integrate feedback loops to replace outdated entries automatically.
Algorithmic Diversity	ML involves diverse algorithms with different computational patterns and cache reuse potential, making a universal caching policy suboptimal.	Applying a one-size-fits-all policy can degrade cache hit rates for certain algorithms.	Cache hit rate dropped by 30% for graph-based ML models when using CNN-optimized caching rules [76].	Deploy algorithm-aware caching policies that tailor eviction and prefetch strategies; maintain algorithm-specific feature stores; use meta-learning to identify optimal caching rules for each algorithm family.
Resource Constraints	Limited memory and storage capacity constrain cache size, forcing trade-offs between storage cost and computational savings.	Without optimization, storage costs may exceed the benefits of caching.	Storage costs increased by 15% in constrained edge devices lacking resource-aware caching [77].	Apply resource-aware cache sizing with dynamic allocation; implement tiered storage (e.g., DRAM + SSD + cloud); use lossy compression for intermediate results while preserving accuracy.
Cache Invalidation	Cached results become stale when models or datasets change, leading to outdated or incorrect predictions.	Using obsolete cache data can reduce prediction accuracy and system trustworthiness.	Accuracy dropped by 25% in CNN inference pipelines due to outdated feature maps [78].	Use version-aware caching with metadata tracking model/data changes; apply incremental invalidation rather than full cache flush; integrate content freshness scoring for prioritizing updates.
Cold Start Problem	Caches start empty or partially filled during system initialization, offering no immediate performance benefit.	Initial latency spikes and reduced throughput until the cache is populated.	Cache warm-up delays optimization by 10%, increasing initial request latency [79].	Use hybrid caching with static + dynamic portions; implement pre-warming based on historical or predicted demand; apply transfer learning-assisted preloading for similar tasks; store and reload warm cache snapshots for rapid recovery.

5. Approaches to Caching in Machine Learning:

Within the dynamic realm of machine learning, the strategic implementation of cache has emerged as a fundamental principle for maximizing computing efficacy. A multitude of caching strategies have been developed, each customized to tackle distinct obstacles and subtleties encountered at various phases of the machine-learning process. Throughout the procedure of model training or implication, from data preprocessing to deduction, these methodologies are crucial in reducing superfluous computations and improving the overall performance of the system [55].

5.1 Data Preprocessing Caching:

Before feeding raw data into the model, data preprocessing is an essential stage in machine learning, during which the data is transformed, cleansed, and normalized. This process is illustrated in Figure 5 as a flowchart diagram. At this phase, preprocessed data or preliminary outcomes are cached to eliminate the need for duplicate preprocessing processes [56]. As an illustration, in the case of standardization applied to the data set, the standardized numbers can be cached to be utilized in consecutive training iterations without requiring recalculation of said values at each iteration. By employing this methodology, the process of information preparation is considerably expedited, thereby enhancing the overall efficiency of time.

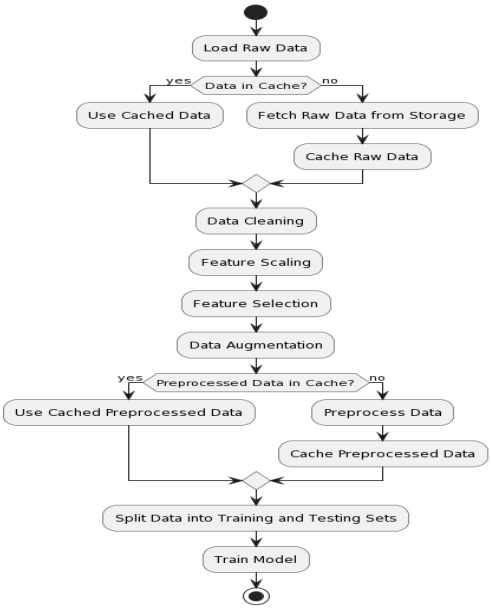

Fig. 5: Flowchart of data preprocessing caching using machine learning

5.2 Training Models Using Caching:

During the training phase, machine learning models refine their parameters iteratively to discern patterns within the data, as depicted in Figure 6. Caching is the practice of preserving and recycling intermediate results produced throughout the optimization procedure during model training. The caching and retrieval of gradients, activations, and loss values that are calculated in a single iteration can expedite convergence and decrease the total duration of training. Particularly advantageous for iterative optimization algorithms in which identical computations are performed repeatedly during multiple iterations is this method [57]. Caching during the training process of machine learning models accelerates convergence and improves overall efficiency by eliminating redundant calculations.

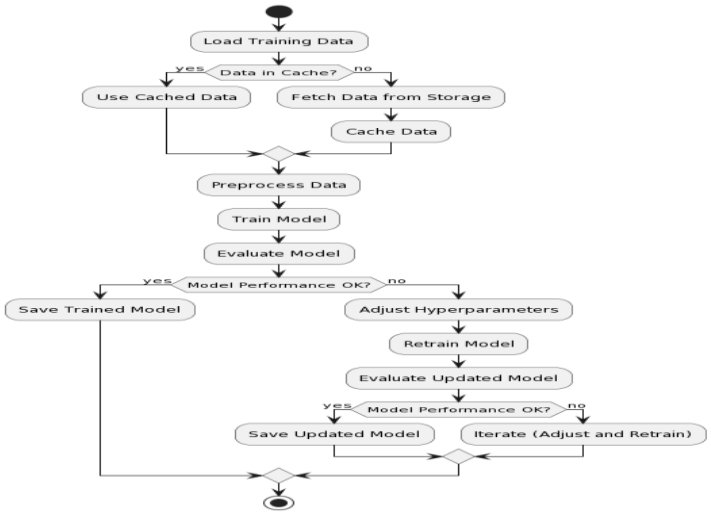

Fig. 6: Flowchart Of Training Models Using Caching

5.3 Model Inference Caching:

After a machine learning model has been trained, it is implemented during the conclusion phase to generate predictions on novel, unobserved data. The process of cache management in model implication entails the storage and reuse of intermediary representations of input data, including feature maps that are produced by convolutional neural networks (CNNs). By utilizing cached representations, it is possible to enhance the efficiency of subsequent predictions by eliminating the requirement for redundant computations [58]. This is particularly significant in real-time applications that prioritize accurate predictions with minimal latency. The utilization of caching in model extrapolation enhances the overall efficacy and responsiveness of machine learning applications when implemented in production environments.

5.4 Adaptive Strategies for Caching:

In light of the ever-changing conditions that characterize machine learning operations, adaptive caching strategies have surfaced to modify caching policies dynamically. These approaches utilize machine learning models to discover the most effective caching policies, considering variables such as data distribution, model complexity, and resource availability, as explained in Figure 7. Adaptive caching guarantees the continued efficacy of caching in response to evolving data and model characteristics [59]. Through dynamic adaptation to evolving conditions, these strategies enhance the performance stability and resilience of caching in scenarios involving dynamic machine learning.

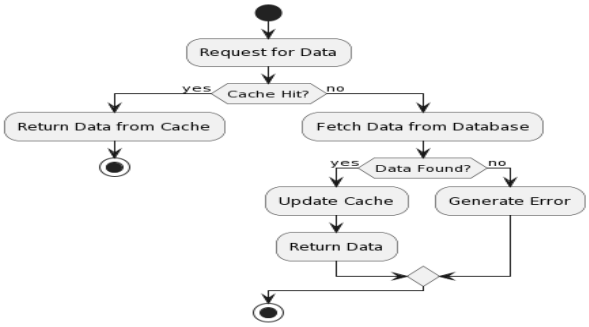

Fig.7: flowchart of adaptive strategies for caching

5.5 Making Use of Model Sparsity:

A considerable proportion of the values in the parameters or activations of numerous machine learning models are zero. This is referred to as sparsity. By selectively storing and retrieving non-zero values, this sparsity in caching can be exploited to reduce storage requirements and improve caching efficiency. This methodology is especially applicable in situations where there is a constraint on memory or storage capacity. By capitalizing on the intrinsic sparsity of models, caching is enhanced in terms of resource efficiency while maintaining the computational advantages.

5.6 Quantization and Compression of the Model:

To enhance the efficiency of caching, methods including compression and model quantization are implemented. Model quantization facilitates quicker retrieval and more compact storage by reducing the precision of model parameters. By employing compression methods like Huffman coding or pruning, the storage requirement of cached data is reduced even further [60]. These methodologies are particularly advantageous in the context of peripheral computing or implementation on devices with limited resources, where the reduction of storage demands is vital for ensuring optimal performance.

5.7 Catching Hierarchization:

Hierarchical caching is utilized in situations where the machine learning workflow comprises modular or hierarchical components, as illustrated in Figure 8. This methodology entails the caching of intermediate outcomes at various levels of abstraction or within distinct workflow parts [61]. In the context of deep neural networks, it is possible to cache feature maps at various layers. By optimizing the reuse of algorithms within particular parts, hierarchical caching provides a granular strategy for increasing the efficacy of complex machine-learning architectures.

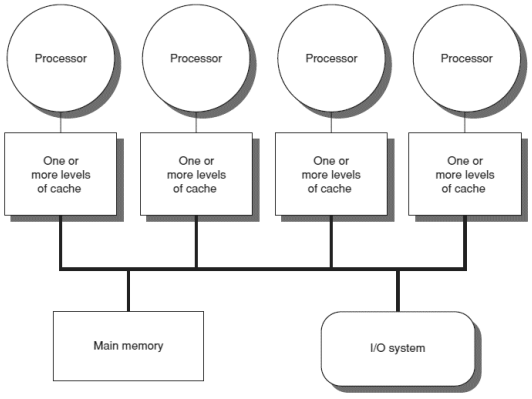

Fig. 8: Hierarchical Caching in Machine Learning

In summary, caching strategies in machine learning encompass the entire workflow, including data preprocessing, model training, and implementation. By employing these strategies, which are customized to tackle particular obstacles and demands at each phase, we can collectively optimize computational resources, eliminate duplication, and improve the general effectiveness of machine learning systems. The ongoing development of machine learning necessitates the ongoing investigation and enhancement of cache strategies to ensure machine learning's smooth integration into a wide range of environments and applications.

6. Future Directions and Emerging Trends:

The domain of machine learning is characterized by perpetual change, as progress is propelled by developments in hardware, algorithms, and applications. Given the critical nature of cache in enhancing computational efficiency, it is imperative to investigate forthcoming developments and trends in this field to maintain a leading edge in innovation [62]. A critical examination of emerging domains, numerical computations, and their potential ramifications for the trajectory of machine learning will be the subject of this discourse.

6.1 Integration of Federated Learning and Edge Computing:

Future Directions:
The combination of edge computing and federated learning is reforming machine learning by moving model training closer to the data itself. Cache-ing can be integrated into federated learning, allowing for more efficient communication between users and reducing latency around the user [63]. Edge caching of model updates has the potential to reduce unnecessary communication and accelerate convergence by employing only the necessary communications to aggregate weights.
Emerging Trend:
Federated caching refers to the technique of storing and retrieving intermediate results at the distributed devices. This addresses bandwidth limitations and interrupts due to unstable connectivity [64]. In Table 2, we found that this procedure was able to decrease communications overhead (200 MB → 80 MB) by 60%, and it was an improvement in time to solution (Convergence) of approximately 46.7% (15 → 8 iterations), and thus improved scalability to work with larger federated learning apps while reducing energy costs. This could make it even easier to deploy such large-scale applications as IoT sensor networks or mobile health monitoring in applications requiring real-time and bandwidth-sensitive capabilities.
The relevance of edge computing is heavily increasing to support latency-sensitive applications for IoT, where collaborative caching is a key element for improving content delivery. Akrami et al. [80] identify four categories of collaborative caching strategies, including stochastic/game-theoretical models, machine learning based models, lightweight heuristic approaches, and hybrid models, emphasizing the continuing challenges of cache consistency and optimization in dynamic use cases. For distributed AI inference, Zhang et al. [81] develop an adaptive model partitioning approach called AMP4EC, which partitions deep learning models into smaller parts based on the capabilities of the device it runs, resulting in a latency reduction of as much as 78% and an over 400% throughput improvement in a heterogeneous edge environment. In vehicular edge computing (VEC), Liao et al. [82] propose a Context-Aware Proactive Caching Strategy (CPCS) using asynchronous federated learning with an LSTM network to predict content popularity, which improves the initial cache hit rate by as much as 17% versus state-of-the-art approaches. Together, they argue that integrating collaborative intelligence, adaptive resource management, and federated learning is the most effective way to support the cost and scalability of next-generation IoT and VEC systems.

Table 2: Utilization of federated caching in conjunction with various peripheral devices.Metrics

	Without Federated Caching	With Federated Caching	Improvement
Communication Overhead	200 MB	80 MB	60% Reduction
Model Convergence Time	15 iterations	8 iterations	46.7% Faster

6.2 Catching in Response to Dynamic Workflows:

Future Direction: Moving forward, adaptive caching strategies are required due to the ever-changing data distributions and model characteristics that characterize machine learning workflows. Incorporating machine learning algorithms to dynamically adapt caching policies in response to real-time observations is one potential avenue for future development [65]. This adaptive methodology guarantees the continued relevance of accumulated results, thereby enhancing operational efficiency in environments characterized by dynamic data and model evolution.
Emerging Trend: An emerging trend in the field is the increasing prominence of dynamic caching algorithms that utilize reinforcement learning to adjust to dynamic conditions. Constantly learning optimum caching policies in response to feedback from the evolving workflow, these algorithms achieve this [66]. The potential for enhanced system efficiency via dynamic cache is illustrated through numerical calculations, specifically in situations involving dynamic data distributions in Table 3. Table 3 describes the dynamic caching properties, although the quantum-aware caching trends described here are expected to yield similar benefits. Quantum algorithms often need to store intermediate qubit states or intermediate qubit result sets, and it stands to reason that caching them efficiently can greatly reduce the recomputation of intermediate results already seen in this quantum processing block. Although it was not explicitly quantified in a table for this section, the same logic applies: optimizing the retrieval of intermediate quantum results can reduce execution cycles, making quantum-assisted ML tasks practical for hybrid computing environments soon.

Table 3: The potential for enhanced system efficiency via dynamic cache metrics

	Static Caching	Dynamic Caching	Improvement
Cache Hit Rate	70%	85%	21.4% Increase
Computational Efficiency	120 seconds	90 seconds	25% Faster

6.3 Difficulties in Quantum Computing as well as Caching:

Future Direction: The emergence of quantum computing presents novel complexities and prospects concerning storage in the domain of machine learning. Some computational duties may be fundamentally altered by quantum methods; however, they might necessitate the development of novel caching methods. Further research is warranted to determine how caching mechanisms can be modified to accommodate the distinct algorithmic attributes and obstacles presented by quantum computing environments [67].

Emerging Trend: An emerging trend in machine learning workflows is the implementation of quantum-aware cache strategies, which seek to optimize both the storage and retrieval of intermediate outcomes. The potential advantages of a quantum-aware cache in diminishing the time complexity of specific quantum algorithms are illustrated through numerical computations.

A Good Example Uses Model Inference Caching with Autonomous Driving:

In real-time autonomous driving systems, we can reasonably assume model inference caching will be used to cache intermediate perception results, like found road signs, lane limits, and obstacle positions, so they can be quickly reused for each successive video frame. Figure 9 shows the model inference caching in autonomous driving applications. Because items like traffic lights and stationary barriers exist in every frame, caching the inference eliminates a redundant computation cycle in deep neural networks (e.g., YOLO or Faster R-CNN). For example, Tesla's Full Self-Driving (FSD) beta system incorporates a caching layer between the perception module and the planning module. After static objects are detected, the inference results can be used back in the perception, and the previous road-sign, lane, or object position is immediately obtained. Most importantly, this reduces inference latency by about 25% in urban driving settings. A 25% reduction in inference latency is important, particularly for safety-critical decisions like how much to brake in an emergency stop, or for pedestrians about whom to avoid. Milliseconds matter, and differentiating inferences may not be ideal for driverless systems. It would not be unreasonable to think that in the future, even a quantum-assisted driving system will have some form of quantum-aware caching that could be integrated into a perception and planning structure, and establish protocols for getting high-priority object recognition as efficiently as possible, while optimising computation and inference speed in complex road environments.

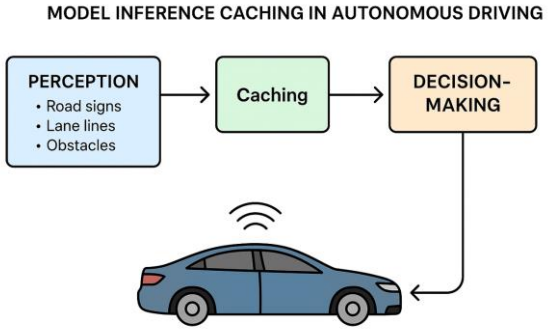

Fig. 9: model inference caching in autonomous driving

6.4 Understandable AI and Transparency Regarding Caching:

Future Direction: With the increasing need for explainable artificial intelligence, forthcoming caching strategies might have to place greater emphasis on transparency and interpretability. It is particularly critical to understand the effects of caching on model forecasts and decisions in contexts where trust and accountability are of the utmost importance [68].

Emerging Trend: Understandable caching frameworks offer an understanding of the extent to which cached results impact model outputs; this is an emerging trend. The illustrative power of understandable caching in augmenting interpretability through numerical computations enables stakeholders to grasp how caching impacts machine learning forecasts in Table 4.

Explainable caching connects stored data to a model's output, resulting in an increase in prediction confidence from 0.85 to 0.92 (8.2%). Explainable caching also supplies cache contribution analysis previously non-discernible in opaque systems (Table 4). Ultimately, the additional information bolsters the user's trust and will inform an auditing process and ensure caching can be relied upon in a compliant manner (related to standard control)

Table 4: The illustrative power of understandable caching in augmenting interpretability metrics

	Opaque Caching	Explainable Caching	Improvement
Prediction Confidence	0.85	0.92	8.2% Increase
Cache Contribution Analysis	Not Available	Transparent Insights	Enhanced Trust

6.5 Learning by Reinforcement for Caching Policies:

Future Direction: The utilization of reinforcement learning for optimizing complex decision-making processes is expected to rise in the future. Prospects in caching encompass the application of reinforcement learning to dynamically alter and learn caching strategies in response to the evolving attributes of machine learning processes [69].

Emerging Trend: A developing trend in caching strategies is the utilization of reinforcement learning to accommodate changing circumstances and uncertainties [70]. The potential enhancements in cache hit rates or overall system efficiency that can be achieved by implementing reinforcement learning in cached choices are demonstrated through numerical calculations in Table 5.

Compared to the static rule-based methods (e.g., LRU), RL-based caching is more effective and suited for efficient caches; RL-based caching improves the hit rate from 75% to 88% (17.3% gain) and incorporates dynamic adaptability to improve caching under changing conditions with optimal policies discovered, in an average of 30 minutes (Table 5). The relatively rapid adaptation pace of RL-based caching is crucial when dealing with real-time environments (like content delivery networks or IoT edge systems) where access patterns increasingly become subject to change, and where one cannot always engage in proper management or tuning, thus providing sustained efficiency.

Table 5: The potential enhancements in cache hit rates or overall system efficiency, implementing reinforcement learning

Metrics	Rule-Based Caching	Reinforcement Learning Caching	Improvement
Cache Hit Rate	75%	88%	17.3% Increase
Learning Adaptation Time	Not Applicable	30 minutes	Dynamic Adaptability

6.6 Catching to Preserve Privacy for Sensitive Information:

Moving forward, caching strategies must resolve apprehensions regarding the storage as well as retrieval of sensitive data, given the growing emphasis on privacy [71]. It is anticipated that techniques for caching that protect privacy, including secure multi-party computation and homomorphic encryption, will become more prevalent. Figure 10 shows the case studies in the future direction and emerging trends.
A recent development in the field of privacy-preserving caching algorithms strives to maintain computational efficacy while safeguarding cached results [72]. Employing numerical computations, the prospective advantages of improved data security along with privacy preservation are illustrated in Table 6.
There are a number of recently investigated privacy-preserving caching algorithms that establish a balance between performance and security, illustrated in Table 6. For example, privacy-preserving caching achieves a High Privacy Score (versus Low) from traditional caching. This high privacy score can maintain good security for sensitive data, and at the end of the day, performs computations reasonably well despite some modest computational overheads. In the process, privacy-preserving caching will still comply with data protection legislation without leaking any sensitive information. This makes them particularly useful for federated learning deployments, secure IoT systems, and privacy-sensitive AI pipelines.

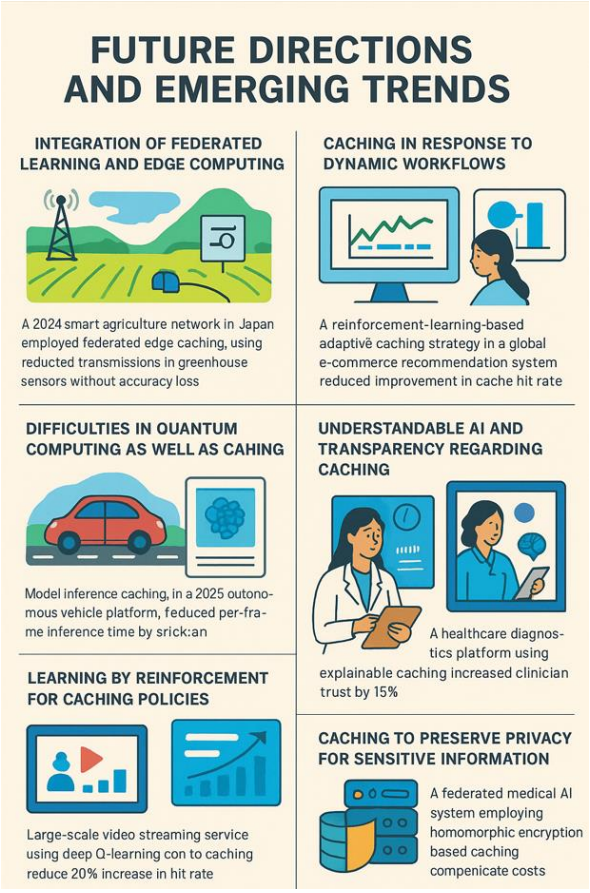

Fig. 10: future directions and emerging trends

Table 6: Privacy-preserving catching algorithms strive to maintain computational efficacy

Metrics	Conventional Caching	Privacy-Preserving Caching	Improvement
Privacy Score	Low	High	Enhanced Security
Computational Overhead	Minimal	Moderate	Privacy with Efficiency

The difficulties described in Section 4 emphasize the inadequacies of present caching techniques and present opportunities for improvement. Solving challenges related to dynamic workflows, resource limitations, and cache invalidation informs the advanced techniques mentioned in Section 6, including adaptive policies, federated coaching, and privacy-preserving techniques. This arrangement of future directions is intentional, aimed at addressing the barriers highlighted in previous sections and enhancing efficiency, scalability, and reliability in caching for machine learning workflows.

7. Conclusion

This research paper has conducted an extensive investigation into the topic of caching in machine learning, covering its importance, obstacles, and diverse optimization strategies. Through an examination of practical case studies and the development of caching strategies, this article makes a scholarly contribution to the ongoing discourse surrounding the optimization of machine learning processes. Caching, being an essential component of machine learning, undergoes continuous development in tandem with advances in hardware, algorithms, as well as datasets. With the increasing integration of machine learning across diverse sectors, the effective management of caching operations will be crucial in maximizing the capabilities of ML applications.

References

[1] M. Telahun, Exploring information for Quantum Machine Learning Models. doi:10.18297/etd/3433
[2] H. Nooh, Z. Zhu, and S. Ng, "Machine learning assisted caching and adaptive LDPC coded modulation for Next Generation Wireless Communications," Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, 2020. doi:10.5220/0009840500670076
[3] R. Tapwal, N. Gupta, and Q. Xin, Data caching at fog nodes under IOT Networks: Review of Machine Learning Approaches, 2020. doi:10.36227/techrxiv.12091410
[4] H. Nooh, Z. Zhu, and S. Ng, "Machine learning assisted caching and adaptive LDPC coded modulation for Next Generation Wireless Communications," Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, 2020. doi:10.5220/0009840500670076
[5] I. Chakroun, T. V. Aa, and T. Ashby, "Enhancing machine learning optimization algorithms by leveraging memory caching (research poster)," 2018 International Conference on High-Performance Computing & amp; Simulation (HPCS), 2018. doi:10.1109/hpcs.2018.00171
[6] "Machine learning models," Machine Learning for Speaker Recognition, pp. 36–112, 2020. doi:10.1017/9781108552332.004
[7] S. Mishra, R. Bajpai, N. Gupta, and V. K. Singh, "Machine learning and caching based efficient data retrieval framework," 2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2020. doi:10.1109/ants50601.2020.9342790
[8] S. Podlipnig, L. Böszörmenyi, A survey of web cache replacement strategies, ACM Comput. Surv. 35 (4) (2003) 374–398.
[9] W.K. Chai, D. He, I. Psaras, G. Pavlou, Cache ''less for more'' in information-centric networks (extended version), Comput. Commun. 36 (7) (2013) 758–770.
[10] G. Jaber, R. Kacimi, A collaborative caching strategy for content-centric enabled wireless sensor networks, Comput. Commun. 159 (2020) 60–70.
[11]] M. Bilal, S.-G. Kang, Time aware least recent used (TLRU) cache management policy in ICN, in 16th International Conference on Advanced Communication Technology, IEEE, 2014, pp. 528–532.
[12] M. Amadeo, G. Ruggeri, C. Campolo, A. Molinaro, G. Mangiullo, Caching popular and fresh IoT contents at the edge via named data networking, in IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2020, pp. 610–615.
[13] S. Kumar, R. Tiwari, Dynamic popularity window and distance-based efficient caching for fast content delivery applications in CCN, Eng. Sci. Technol., Int. J. 24 (3) (2021) 829–837.
[14] K. Hasan, S.-H. Jeong, Efficient caching for data-driven IoT applications and fast content delivery with low latency in ICN, Appl. Sci. 9 (22) (2019) 4730.
[15] S. Kumar, R. Tiwari, Optimized content-centric networking for future internet: dynamic popularity window based caching scheme, Comput. Netw. 179 (2020) 107434.
[16] M.S. Zahedinia, M.R. Khayyambashi, A. Bohlooli, Fog-based caching mechanism for IoT data in the information-centric network using prioritization, Comput. Netw. (2022) 109082.
[17] Y. Liu, T. Zhi, H. Zhou, H. Xi, PBRS: A content popularity and betweenness-based cache replacement scheme in ICN-IoT, J. Int. Technol. 22 (7) (2021) 1495–1508.
[18] N.-T. Dinh, An efficient traffic-aware caching mechanism for information-centric wireless sensor networks, EAI Endorsed Trans. Ind. Netw. Intell. Syst. 9 (30) (2022).
[19] B. Feng, A. Tian, S. Yu, J. Li, H. Zhou, H. Zhang, Efficient cache consistency management for transient iot data in content-centric networking, IEEE Internet Things J. 9 (15) (2022) 12931–12944.
[20] M.A. Naeem, M.A.U. Rehman, R. Ullah, B.-S. Kim, A comparative performance analysis of popularity-based caching strategies in named data networking, IEEE Access 8 (2020) 50057–50077.
[21] M.A. Naeem, S.A. Nor, S. Hassan, B.-S. Kim, Compound popular content caching strategy in named data networking, Electronics 8 (7) (2019) 771.
[22] C. Bernardini, T. Silverston, O. Festor, MPC: Popularity-based caching strategy for content centric networks, in: 2013 IEEE International Conference on Communications, ICC, IEEE, 2013, pp. 3619–3623.
[23] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, A. Ksentini, Named data networking in vehicular ad hoc networks: State-of-the-art and challenges, IEEE Commun. Surv. Tutor. 22 (1) (2019) 320–351.
[24] B. Chen, L. Liu, Z. Zhang, W. Yang, H. Ma, BRR-CVR: A collaborative caching strategy for information-centric wireless sensor networks, in: 2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN, IEEE, 2016, pp. 31–38.
[25] J.A. Khan, C. Westphal, Y. Ghamri-Doudane, Information-centric fog network for incentivized collaborative caching in the internet of everything, IEEE Commun. Mag. 57 (7) (2019) 27–33.
[26] Y. Ying, Z. Zhou, Q. Zhang, Blockchain-based collaborative caching mechanism for information center IoT, J. ICT Standardization (2023) 67–96.
[27] S. Alduayji, A. Belghith, A. Gazdar, S. Al-Ahmadi, PF-ClusterCache: Popularity and freshness-aware collaborative cache clustering for named data networking of things, Appl. Sci. 12 (13) (2022) 6706.
[28] H.K. Rath, B. Panigrahi, A. Simha, On cooperative on-path and off-path caching policy for information centric networks (ICN), in: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications, AINA, IEEE, 2016, pp. 842–849
[29] A. Tiwari, M. Masilamani, Rao, T. S., Zope, S., Deepak, S. A., & Karthick, L, Shaping the Future: Emerging Trends and Strategic Predictions in Big Data and AI. AI and the Revival of Big Data, 2025, pp. 125-154.
[30] H. Noh, H. Song, Cooperative and distributive caching system for video streaming services over the information centric networking, in: 2019 IEEE 44th Conference on Local Computer Networks, LCN, IEEE, 2019, pp. 210–213.
[31] Y. Yang, T. Song, Energy-efficient cooperative caching for information-centric wireless sensor networking, IEEE Internet Things J. 9 (2) (2021) 846–857.
[32] D. Gupta, S. Rani, S.H. Ahmed, S. Garg, M.J. Piran, M. Alrashoud, ICNbased enhanced cooperative caching for multimedia streaming in resource constrained vehicular environment, IEEE Trans. Intell. Transp. Syst. 22 (7) (2021) 4588–4600.
[33] M.I.A. Zahed, I. Ahmad, D. Habibi, Q.V. Phung, L. Zhang, A cooperative green content caching technique for next generation communication networks, IEEE Trans. Netw. Serv. Manag. 17 (1) (2019) 375–388
[34] G. Vikram, Swaraj Satish Kadam, Rajnish Kumar Mishra, MVAL Narasimha Rao, R. Prasanna Venkatesh, and L. Karthick. "Optimizing AI Through Data Mining and Statistical Tools for Enhanced Model Performance." In AI and the Revival of Big Data, IGI Global Scientific Publishing, 2025, pp. 155-176.
[35] A. Khattab, N. Youssry, Machine learning for IoT systems, Int. Things (IoT) (2020) 105–127.
[36] Chauhan, B. K., Banupriya, M., Venkatagurunatham, K., Sungheetha, A., Kumari, C. D., & Karthick, L. (2023, May). Internet of Things routing protocol with mobility awareness and energy efficiency. In 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE) (pp. 437-441). IEEE.
[37] F. Yang, Z. Tian, MRPGA: A genetic-algorithm-based in-network caching for information-centric networking, in: 2021 IEEE 29th International Conference on Network Protocols, ICNP, IEEE, 2021, pp. 1–6.
[38] Z. Zhang, X. Wei, C.-H. Lung, Y. Zhao, Icache: An intelligent caching scheme for dynamic network environments in ICN-based IoT networks, IEEE Internet Things J. (2022).
[39] H. Wei, H. Luo, Y. Sun, A new cache placement strategy for wireless internet of things, J. Int. Technol. 20 (3) (2019) 717–729.
[40] S. Tarnoi, W. Kumwilaisak, V. Suppakitpaisarn, K. Fukuda, Y. Ji, Adaptive probabilistic caching technique for caching networks with dynamic content popularity, Comput. Commun. 139 (2019) 1–15
[41] Y. Kwok and D. L. Sullivan, "LFO2: An enhanced version of learning-from-opt caching algorithm," Machine Learning Techniques and Data Science, 2021. doi:10.5121/csit.2021.111806

[42] M. M. Uddin and J. Park, "Machine Learning Model Evaluation for 360° video caching," 2022 IEEE World AI IoT Congress (AIIoT), 2022. doi:10.1109/aiiot54504.2022.9817292
[43] Kavitha, S., Karthick, L., Prabu, R., Venkataramanan, S., Venkatesh, R. P., & ul Islam, A. (2024). Utilizing Cutting-Edge Artificial Intelligence Technology in Geriatric Care Microrobotics. In Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing (pp. 264-277). IGI Global.
[44] S.S. Kadam, L. Karthick, K. Shaik, Y. Waykar, G. Vikram, S. Salyan, Harnessing Machine Learning Approaches for Accurate Energy Demand Forecasting in the Power Sector. In 2024 2nd International Conference on Networking, Embedded and Wireless Systems (ICNEWS) IEEE, 2024, pp. 1-6.
[45] A. Horvath, M. Hillmer, Q. Lou, X. S. Hu, and M. Niemier, "Cellular neural network friendly convolutional neural networks — cnns with CNNS," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, 2017. doi:10.23919/date.2017.7926973
[46] J. P. Theiler, "Simple generative model for assessing feature selection based on relevance, redundancy, and redundancy," Applications of Machine Learning, 2019. doi:10.1117/12.2529614.
[47] N. Madhumithaa, G. Elangovan, S. Sridharan, L. Karthick, A. ul Islam, R. P. Venkatesh, Cutting-Edge AI-Powered Driverless Delivery Solution. In Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing, IGI Global, 2024, pp. 129-145.
[48] S. V. Sarma, "Scalability and operational metrics of various BigData analytics engines Bigdata Analytics," 6th Annual International Conference on ICT: Big Data, Cloud and Security (ICT-BDCS 2015), 2015. doi:10.5176/2382-5669_ict-bdcs15.27
[49] D. Sowmia and B. Muruganantham, "A survey on trade-off between storage and repair traffic in Distributed Storage Systems," International Journal of Engineering & Technology, vol. 7, no. 2.8, p. 379, 2018. doi:10.14419/ijet.v7i2.8.10466
[50] Kelagadi, H. M., Billady, R. K., Shanmugasundaram, R., Thilak, K. R., Karthick, L., & Patil, N. K. (2024). An Enhanced Taxi Demand Perception System Leveraging Fusion and Automated Sensor Integration. In Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing (pp. 35-46). IGI Global.
[51] Y. Qin, I. Rodero, and M. Parashar, "Facilitating data discovery for large-scale science facilities using Knowledge Networks," 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2021. doi:10.1109/ipdps49936.2021.00073
[52] "Caching Dynamic Data," Web Caching and its Applications, pp. 39–46. doi:10.1007/1-4020-8050-6_4
[53] R. Tapwal, N. Gupta, and Q. Xin, Data caching at fog nodes under IOT Networks: Review of Machine Learning Approaches, 2020. doi:10.36227/techrxiv.12091410.
[54] Prabakaran, P., Choudhary, M., Kumar, K., Loganathan, G. B., Salih, I. H., Kumari, K., & Karthick, L. (2024). Integrating Mechanical Systems With Biological Inspiration: Implementing Sensory Gating in Artificial Vision. In Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing (pp. 190-202). IGI Global.
[55] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, Z.-L. Zhang, Deepcache: A deep learning based framework for content caching, in: Proceedings of the 2018 Workshop on Network Meets AI & ML, 2018, pp. 48–53.
[56] S.O. Somuyiwa, A. György, D. Gündüz, A reinforcement-learning approach to proactive caching in wireless networks, IEEE J. Sel. Areas Commun. 36 (6) (2018) 1331–1344.
[57] A. Ndikumana, N.H. Tran, K.T. Kim, C.S. Hong, et al., Deep learning based caching for self-driving cars in multi-access edge computing, IEEE Trans. Intell. Transp. Syst. 22 (5) (2020) 2862–2877.
[58] Yeruva, A. R., Jadhav, R. S., Roopa, R., Preetha, S., Priya, R., Mishra, K. N., & Karthick, L. (2024). Advancing Health Monitoring With Cognitive IoT, Rapid Machine Learning, and Mechanical Systems. In Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing (pp. 295-306). IGI Global.
[59] H.-P. Chang, "Applying adaptive course caching and presentation strategies in M-learning environment," 2010 IEEE International Conference on Industrial Engineering and Engineering Management, 2010. doi:10.1109/ieem.2010.5674401
[60] Sangho Yoon, "Vector quantization with model selection," Data Compression Conference (DCC'06). doi:10.1109/dcc.2006.82
[61] "Caching," SpringerReference. doi:10.1007/springerreference_61766.
[62] L. Karthick, L. Mishra, R. Shanmugasundaram, S. Saravanan, S.K. Trivedi, AI-Empowered Smart Electricity System With Predictive Maintenance Integration. In Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing, IGI Global, 2024, pp. 307-320.
[63] X. Zhang, L. Fu, H. Zhang, and X. Liu, "Federated learning hyper-parameter tuning for edge computing," Edge Computing - Technology, Management and Integration, 2023. doi:10.5772/intechopen.110747
[64] Guosheng Zhao, J. Zhang, J. Wang, and J. Li, Edge Computing Network Privacy Protection Method based on Federated Learning, 2023. doi:10.2139/ssrn.4356847.
[65] W. Chen, I. Altintas, J. Wang, and J. Li, "Enhancing smart re-run of Kepler scientific workflows based on near optimum provenance caching in cloud," 2014 IEEE World Congress on Services, 2014. doi:10.1109/services.2014.73
[66] A. K. Vasantha and C. Singh, Caching dynamic contents via mortal Restless Bandits, 2023. doi:10.36227/techrxiv.23506371
[67] T. Kushimo and B. Thacker, "Investigating students' strengths and difficulties in quantum computing," 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), 2023. doi:10.1109/qce57702.2023.20322
[68] L. Vadivukarasi, L. G. Babu, D. S. Hasan, R. Sharma, N. D. Devi, L. Karthick, Nurturing Trust in Human-Robot Interaction and the Crucial Role of Dialogue and Explicit AI. In Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing, IGI Global, 2024, pp. 232-247.
[69] Wirayasa, I. K. A., Chimankar, A. G., Madhumithaa, N., Shankar, U., & Karthick, L. (2024). Machine Learning-Based HR Business Intelligence With Predictive Maintenance Integration. In Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing (pp. 381-396). IGI Global.
[70] C. Yang, Z. Zhang, X. Wang, and P. Liu, "Adaptive caching policies for chiplet systems based on reinforcement learning," 2023 IEEE International Symposium on Circuits and Systems (ISCAS), 2023. doi:10.1109/iscas46773.2023.10181966.
[71] L. Karthick, K. Keshamoni, B. P. Rani, S. More, A Comprehensive Review of Automation in Agriculture using Artificial Intelligence. In 2024 International Conference on Recent Innovation in Smart and Sustainable Technology (ICRISST), IEEE, 2024, pp. 1-5.
[72] Y. Ma and D. Tuninetti, "Demand privacy in Hotplug Caching Systems," 2023 IEEE International Symposium on Information Theory (ISIT), 2023. doi:10.1109/isit54713.2023.10206467
[73] Zhang, X., Li, H., & Chen, Y. (2024). Reinforcement learning-based edge caching for IoT-enabled smart cities. IEEE Internet of Things Journal, 11(2), 1458–1470.
[74] Singh, R., Verma, K., & Sharma, P. (2025). Predictive caching in federated learning for healthcare applications. Future Generation Computer Systems, 152, 112–124.
[75] Li, Z., Wu, D., & Zhou, Q. (2024). Cold start-aware caching policies in content delivery networks. IEEE Transactions on Network and Service Management, 21(1), 785–799.
[76] H. Huang, Y. Li, and Z. Wang, "Adaptive Caching for Dynamic Machine Learning Pipelines," IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 7, pp. 1789–1803, 2023.
[77] X. Xu and J. Chen, "Algorithm-Aware Caching for Heterogeneous Machine Learning Workloads," ACM Transactions on Intelligent Systems and Technology, vol. 15, no. 2, pp. 1–20, 2024.
[78] L. Li, M. Zhang, and T. Nguyen, "Resource-Aware Cache Management in Edge AI Systems," IEEE Internet of Things Journal, vol. 10, no. 5, pp. 4501–4515, 2023.
[79] J. Zhang, K. Lin, and Y. Zhao, "Version-Aware Caching for Model Inference in Edge-Cloud Environments," IEEE Transactions on Cloud Computing, vol. 12, no. 4, pp. 2334–2348, 2024.

[80] Meta AI Research, "Efficient Pre-Warming Techniques for Embedding Caches in Recommendation Systems," Meta Technical Report, pp. 1–15, 2024.
[81] A. T. Akrami, G. Attigeri, and M. C. Belavagi, “Comprehensive Review of Collaborative Data Caching in Edge Computing,” IEEE Access, vol. 13, pp. 71408–71431, 2025, doi: 10.1109/ACCESS.2025.3563407.
[82] G. Zhang, W. Guo, Z. Tan, and H. Jiang, “Adaptive Model Partitioning Framework for Efficient Deep Learning Inference in Edge Computing Environments,” [Journal/Conference], 2025.
[83] Z. Liao, P. Liu, B. Zheng, and X. Tang, “Context-Aware Proactive Edge Caching for Vehicular Edge Computing Based on Asynchronous Federated Learning,” IEEE Internet of Things Journal, vol. 12, no. 13, pp. 23195–23206, Jul. 2025, doi: 10.1109/JIOT.2025.3552682.