iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84254

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

Artificial Intelligence-Based Predictive Approximation of Belady's Optimal Page Replacement Algorithm Using Deep Learning and Reinforcement Learning

Mr. Anshid Babu T M[1], Mr. Arjun T[2]
[1]*Assistant Professor Department of Computer Science , SAFI Institute of Advanced Study(Autonomous) , Malappuram , India.*
[2]*Assistant Professor Department of Computer Applications, Sree Narayana College Vatakara, Kozhikode*

Abstract: ***Efficient memory management is one of the fundamental responsibilities of modern operating systems, with page replacement algorithms playing a significant role in reducing page faults and improving overall system performance. Among the existing page replacement techniques, Belady's Optimal (OPT) algorithm provides the minimum achievable page fault rate by replacing the page whose next reference occurs farthest in the future. Although OPT represents the theoretical upper bound for page replacement performance, its practical implementation is impossible because future memory references are unknown during program execution [1]. Consequently, contemporary operating systems employ heuristic algorithms such as First-In-First-Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), Clock, and Adaptive Replacement Cache (ARC), which utilize historical memory access information to approximate optimal replacement decisions [2]–[6].***

Recent advances in Artificial Intelligence (AI) have enabled new approaches for predicting sequential data, making predictive memory management increasingly feasible. Deep learning architectures, particularly Long Short-Term Memory (LSTM) networks, effectively learn temporal dependencies in sequential data, while Reinforcement Learning (RL) enables adaptive decision-making through continuous interaction with the execution environment [7]–[10]. These capabilities provide an opportunity to estimate future memory references rather than assuming perfect future knowledge.

This paper proposes an Artificial Intelligence-Based Predictive Optimal Page Replacement (AI-POPR) framework that integrates an LSTM-based prediction engine with a reinforcement learning agent to approximate the behavior of Belady's Optimal algorithm. The framework analyzes historical memory reference sequences, predicts forthcoming page accesses, and dynamically selects victim pages using a hybrid decision strategy. Experimental evaluation using synthetic and benchmark workloads demonstrates that the proposed framework significantly reduces page faults and improves hit ratio compared with conventional algorithms while closely approximating the theoretical performance of OPT. The proposed AI-POPR framework provides an adaptive, intelligent, and practically deployable solution for next-generation operating systems, cloud computing platforms, virtualization environments, and edge computing infrastructures.

Keywords: Artificial Intelligence, Belady's Optimal Algorithm, Deep Learning, Long Short-Term Memory,Machine Learning, Operating Systems, Page Replacement, Predictive Memory Management, Reinforcement Learning, Virtual Memory.

I. INTRODUCTION

A. Background

The rapid expansion of cloud computing, virtualization, artificial intelligence applications, big data analytics, and high-performance computing has substantially increased the demand for efficient memory management within modern operating systems. Virtual memory provides an abstraction that allows processes to execute even when the total memory requirement exceeds the available physical memory by temporarily storing inactive pages on secondary storage [2], [3]. This mechanism enables higher system utilization, increased multiprogramming capability, and improved application scalability.

Demand paging is one of the most widely adopted virtual memory techniques in contemporary operating systems. Under this approach, memory pages are loaded into physical memory only when they are first referenced by the executing process. Although demand paging improves memory utilization, every reference to a page not currently resident in main memory generates a page fault. Because secondary storage access is significantly slower than main memory access, excessive page faults considerably degrade system performance by increasing execution time and processor idle periods [4]–[6].

The responsibility of selecting an appropriate page for replacement lies with the page replacement algorithm. An efficient replacement policy minimizes page faults while maintaining low computational overhead and high memory utilization. Over the past several decades, numerous page replacement algorithms have been proposed, each exploiting different characteristics of memory access behavior.

First-In-First-Out (FIFO) replaces the oldest page in memory regardless of its future usage and is simple to implement. However, FIFO may exhibit Belady's anomaly, where increasing the number of available page frames unexpectedly increases the number of page faults [1]. Least Recently Used (LRU) improves replacement performance by exploiting temporal locality, assuming that recently accessed pages are more likely to be referenced again. Least Frequently Used (LFU) utilizes historical access frequency, while Clock and Second-Chance algorithms approximate LRU using hardware-maintained reference bits to reduce implementation overhead. More advanced adaptive algorithms, including Adaptive Replacement Cache (ARC) and Clock with Adaptive Replacement (CAR), dynamically balance recency and frequency to improve cache efficiency under diverse workloads [4]–[6], [17], [18].

Among all page replacement algorithms, Belady's Optimal (OPT) algorithm remains the theoretical benchmark. The OPT algorithm replaces the page whose next reference occurs farthest in the future, thereby guaranteeing the minimum possible number of page faults for any reference sequence [1]. Unfortunately, the algorithm assumes complete knowledge of future memory references, an assumption that cannot be satisfied during actual program execution. Consequently, OPT serves primarily as a theoretical reference for evaluating practical page replacement algorithms rather than a deployable operating system solution.

Recent developments in Artificial Intelligence have created new opportunities to revisit this long-standing research problem. Machine learning algorithms have demonstrated remarkable performance in sequence prediction, recommendation systems, speech recognition, and time-series forecasting by learning hidden patterns from historical observations [7], [14]. Similarly, memory reference sequences exhibit temporal locality, spatial locality, repetitive execution phases, and long-range dependencies that can be effectively modeled using deep neural networks.

Among deep learning architectures, Long Short-Term Memory (LSTM) networks are particularly suitable for sequential prediction because they overcome the vanishing-gradient problem associated with conventional recurrent neural networks through gated memory cells [8]. LSTM models can retain relevant historical information over extended sequences and accurately predict future events based on previous observations. Reinforcement Learning further enhances adaptive decision-making by enabling an intelligent agent to learn optimal replacement policies through interaction with the operating environment and reward-based optimization [9], [15].

These advances suggest that future memory references need not be known exactly; instead, they can be estimated with high probability using predictive Artificial Intelligence models. Replacing deterministic future knowledge with probabilistic prediction allows operating systems to approximate the behavior of Belady's Optimal algorithm while remaining practically implementable.

The motivation for this research is therefore to bridge the long-standing gap between the theoretical optimality of Belady's algorithm and the practical limitations of existing page replacement techniques. By integrating deep sequential prediction with adaptive reinforcement learning, the proposed framework seeks to develop an intelligent page replacement policy capable of reducing page faults, improving hit ratio, and adapting dynamically to changing workload characteristics across cloud computing, virtualization, edge computing, and high-performance computing environments.

B. Problem Statement

Despite decades of research, existing page replacement algorithms continue to rely on heuristic assumptions regarding memory locality. Their decisions are based exclusively on historical page accesses without considering future execution behavior. Consequently, these algorithms often replace pages that are needed shortly afterward, increasing page faults and reducing overall system performance. Belady's Optimal Algorithm theoretically provides the minimum possible page fault rate by selecting the page whose next reference is farthest in the future. Unfortunately, because future page references are unavailable during runtime, the algorithm cannot be directly implemented in practical operating systems. This gap between theoretical optimality and practical feasibility has remained one of the longest-standing challenges in memory management research.

The emergence of Artificial Intelligence introduces the possibility of estimating future page references from historical memory access patterns. However, existing AI-assisted approaches often employ either sequence prediction or reinforcement learning in isolation, limiting their ability to adapt to diverse workloads while maintaining prediction accuracy. There remains a need for a unified framework that combines predictive modeling with adaptive decision-making to achieve near-optimal page replacement in real-world systems.

C. Research Objectives

The primary objective of this research is to develop a practical approximation of Belady's Optimal Page Replacement Algorithm using Artificial Intelligence techniques. Specifically, this study aims to:

1) Design a predictive memory management framework capable of estimating future page references from historical memory traces.
2) Develop a Long Short-Term Memory (LSTM) model that learns sequential memory access patterns and predicts forthcoming page requests.
3) Integrate reinforcement learning to dynamically select victim pages based on predicted future accesses and current system state.
4) Evaluate the proposed framework against conventional page replacement algorithms including FIFO, LRU, LFU, Clock, and the theoretical OPT algorithm.
5) Analyze computational complexity, prediction accuracy, adaptability, and scalability across diverse workload characteristics.

D. Research Contributions

The major contributions of this paper are summarized as follows:

1) A novel Artificial Intelligence-Based Predictive Optimal Page Replacement (AI-POPR) framework is proposed to approximate Belady's Optimal Algorithm using sequential prediction and adaptive learning.
2) An LSTM-based prediction module is designed to estimate future page references from historical memory access sequences.
3) A reinforcement learning module is integrated with the prediction engine to improve victim page selection under changing workloads.
4) A hybrid decision mechanism combines predicted future references, temporal locality, frequency statistics, and reinforcement learning policies to achieve near-optimal replacement decisions.
5) Comprehensive experimental evaluation demonstrates that the proposed framework significantly reduces page faults and improves hit ratio compared with traditional page replacement algorithms while approaching the theoretical performance of the Optimal algorithm.

II. RELATED WORK

A. Evolution of Page Replacement Algorithms

Efficient page replacement has remained one of the most extensively studied problems in operating system memory management since the introduction of virtual memory. The primary objective of a page replacement algorithm is to minimize page faults by selecting the most appropriate page for eviction whenever physical memory becomes full. Over the past several decades, numerous algorithms have been proposed, each attempting to exploit memory locality and application behavior. Despite these developments, no practical algorithm has consistently matched the theoretical performance of Belady's Optimal (OPT) algorithm because future page references remain unknown during execution.

The earliest page replacement strategy, First-In-First-Out (FIFO), replaces the page that has resided in memory for the longest period. FIFO is attractive because of its simplicity, requiring only a queue structure and constant-time insertion and deletion operations. However, FIFO completely ignores the actual access behavior of applications. Consequently, frequently accessed pages may be removed simply because they entered memory earlier than others. An additional drawback is Belady's Anomaly, where increasing the number of available page frames can unexpectedly increase the number of page faults. This anomaly demonstrates that memory allocation alone cannot guarantee improved performance when the replacement policy fails to consider workload characteristics. To overcome FIFO's limitations, researchers proposed the Least Recently Used (LRU) algorithm. LRU assumes that recently accessed pages are likely to be referenced again in the near future, an assumption based on temporal locality. Under many real-world workloads, LRU performs significantly better than FIFO because it retains frequently reused pages in memory. Nevertheless, maintaining exact recency information requires updating timestamps or linked-list structures on every memory reference, introducing computational overhead. Furthermore, LRU performs poorly for cyclic access patterns that exceed the available memory capacity, where frequently revisited pages may still be unnecessarily evicted.

The Least Frequently Used (LFU) algorithm addresses a different aspect of locality by considering historical access frequency. Pages with the lowest reference counts are selected for replacement under the assumption that frequently accessed pages are more valuable. While LFU performs well for stable workloads, it often struggles with dynamic applications where access patterns change over time. Pages that were heavily used in the past may remain in memory despite becoming irrelevant, reducing adaptability to evolving execution behavior.

Several approximate algorithms have also been developed to balance performance and implementation cost. The Clock algorithm, also known as the Second-Chance algorithm, approximates LRU using a circular queue and a reference bit maintained by the hardware. Pages that have been recently accessed receive a second opportunity before eviction, reducing bookkeeping overhead while maintaining acceptable performance. Enhanced versions such as Enhanced Clock incorporate dirty-bit information to minimize unnecessary disk writes during replacement. Although these methods improve efficiency, they remain heuristic and cannot guarantee optimal decisions across diverse workloads.

Modern operating systems frequently employ adaptive algorithms that dynamically balance recency and frequency. The Adaptive Replacement Cache (ARC) maintains separate lists for recently accessed and frequently accessed pages while automatically adjusting their relative importance according to workload characteristics. Similarly, the Clock with Adaptive Replacement (CAR) algorithm combines the low overhead of Clock with ARC's adaptive behavior. These techniques generally outperform traditional LRU under mixed workloads but require additional metadata management and tuning parameters. Moreover, their decisions continue to depend exclusively on historical observations rather than anticipated future accesses.

B. Belady's Optimal Algorithm

Belady introduced the **Optimal Page Replacement Algorithm (OPT)** as a theoretical benchmark for evaluating page replacement policies. OPT replaces the page whose next reference occurs farthest in the future, thereby guaranteeing the minimum possible number of page faults for a given reference string and memory size. Because no algorithm can outperform OPT under identical conditions, it has long served as the upper performance bound for virtual memory systems.

Despite its theoretical significance, OPT cannot be implemented in practical operating systems because future page references are unavailable at runtime. Modern processors execute instructions dynamically, influenced by user interactions, operating system scheduling, input/output operations, interrupts, compiler optimizations, and concurrent processes. These factors make exact prediction of future memory accesses impossible using conventional deterministic methods.

Consequently, virtually all practical page replacement algorithms represent approximations of OPT. Their primary limitation lies in replacing pages based solely on historical information while ignoring forthcoming execution behavior. Bridging the gap between theoretical optimality and practical implementation has therefore become one of the most enduring research challenges in operating systems.

C. Memory Locality and Access Patterns

The effectiveness of page replacement algorithms largely depends on the locality properties exhibited by application memory accesses. Two fundamental locality principles dominate memory behavior.

Temporal locality states that recently accessed memory locations are likely to be accessed again in the near future. This principle forms the theoretical foundation of LRU and related algorithms.

Spatial locality indicates that memory addresses located near recently accessed locations have an increased probability of future access. Sequential program execution, array traversal, and instruction fetching commonly exhibit strong spatial locality.

Modern applications additionally demonstrate higher-order characteristics, including periodic execution phases, nested loops, recursive function calls, and repetitive computational kernels. These patterns create sequential dependencies extending beyond immediate neighboring references. Traditional page replacement algorithms generally fail to exploit such long-range dependencies because they rely on simple heuristics rather than predictive sequence modeling.

Machine learning methods offer a promising alternative by automatically identifying complex temporal relationships within historical memory traces. Instead of relying solely on manually designed heuristics, learning algorithms infer hidden structures from observed data, enabling prediction of future memory references.

D. Machine Learning in Operating Systems

Recent advances in Artificial Intelligence have significantly influenced operating system research. Machine learning techniques have been successfully applied to CPU scheduling, process migration, resource allocation, energy optimization, cache management, storage systems, and network traffic prediction. Unlike traditional rule-based approaches, machine learning algorithms continuously improve their decisions by learning from observed system behavior. Supervised learning methods construct predictive models using labeled historical datasets. In memory management, supervised models can estimate the probability that a page will be referenced within a future time window based on previous access sequences. Common algorithms include Decision Trees, Random Forests, Support Vector Machines, Gradient Boosting, and Artificial Neural Networks.

Unsupervised learning methods identify hidden structures within unlabeled datasets. Clustering techniques can classify applications exhibiting similar memory behaviors, enabling workload-specific replacement policies. Although these methods reveal useful execution patterns, they generally lack explicit predictive capability.

Deep learning has emerged as a particularly attractive solution because of its ability to model nonlinear relationships and long-range sequential dependencies. Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs) have demonstrated remarkable performance in sequence prediction problems such as natural language processing, speech recognition, and financial forecasting. Since memory references also form sequential data streams, these architectures provide a natural framework for predicting future page accesses.

However, deep learning introduces computational overhead associated with model training and inference. Therefore, practical deployment within operating systems requires lightweight prediction models capable of meeting strict latency constraints.

E. Deep Learning for Memory Access Prediction

Among sequence learning architectures, LSTM networks have received particular attention because they effectively overcome the vanishing-gradient problem associated with conventional recurrent neural networks. Through gated memory cells, LSTM models selectively retain relevant historical information while discarding less useful observations. This capability enables accurate prediction of long-range dependencies present within memory access traces.

Several recent studies have demonstrated that application memory references exhibit repetitive patterns that can be learned through sequence modeling. Instruction execution, iterative loops, recursive algorithms, database queries, and scientific computing workloads frequently generate predictable reference sequences. LSTM networks exploit these characteristics by estimating the probability distribution of forthcoming page references based on previously observed accesses.

Transformer architectures have also attracted growing interest due to their self-attention mechanism, which captures relationships among distant sequence elements without relying on recurrent computation. Compared with LSTM, transformers often achieve higher prediction accuracy for long sequences but require significantly greater computational resources. Consequently, transformer-based approaches remain more suitable for cloud servers and high-performance computing environments than resource-constrained embedded systems. Despite encouraging prediction performance, many existing deep-learning-based page replacement studies terminate after forecasting future references. They seldom integrate prediction with an adaptive decision-making mechanism capable of selecting the optimal victim page under changing workload conditions. This limitation motivates combining predictive models with reinforcement learning, which is discussed in the following section of this review.

III. RESEARCH GAP

The page replacement problem has been extensively investigated for more than five decades, leading to the development of numerous replacement algorithms, including First-In-First-Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), Clock, Adaptive Replacement Cache (ARC), and Clock with Adaptive Replacement (CAR). These algorithms have significantly improved memory management in modern operating systems by exploiting temporal locality, spatial locality, and access frequency. Nevertheless, none of these approaches consistently achieves optimal performance under diverse workload conditions because they rely solely on historical memory access information. Belady's Optimal (OPT) page replacement algorithm remains the theoretical benchmark for evaluating replacement policies. By replacing the page whose next reference occurs farthest in the future, OPT guarantees the minimum possible number of page faults for any reference string. However, the algorithm assumes perfect knowledge of future memory references, making it impossible to implement directly in practical operating systems. Consequently, all existing practical algorithms represent heuristic approximations rather than true optimal solutions. Recent advances in Artificial Intelligence (AI) and Machine Learning (ML) have created new opportunities to revisit this classical problem. Deep learning architectures, particularly Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), and Transformer models, have demonstrated exceptional capability in learning sequential patterns from historical data. These techniques have been successfully applied in domains such as natural language processing, financial forecasting, speech recognition, and network traffic prediction. Since memory references also form sequential data streams with temporal dependencies, AI models provide a promising foundation for predicting future page accesses. Several recent studies have investigated AI-assisted page replacement strategies. Most existing approaches employ supervised learning models to estimate the probability of future page references based on historical access patterns. Other studies formulate page replacement as a reinforcement learning (RL) problem, allowing an agent to learn eviction policies through interaction with simulated environments. Although these approaches demonstrate improvements over conventional algorithms, they exhibit several important limitations.

First, most prediction-based methods rely exclusively on sequence learning while neglecting adaptive decision-making. Even when future references are predicted accurately, selecting the optimal victim page requires consideration of additional factors such as page age, reuse distance, dirty-bit status, frame occupancy, and current workload characteristics. Prediction alone does not guarantee optimal replacement decisions.

Second, reinforcement learning-based approaches frequently operate without an explicit prediction mechanism. The RL agent learns through trial-and-error interactions, which often require extensive training and may converge slowly under highly dynamic workloads. Without predictive information, the learning process becomes more computationally expensive and may fail to approach the theoretical performance of Belady's algorithm.

Third, many existing studies evaluate their methods using small synthetic reference strings or limited benchmark datasets. Such evaluations fail to capture the complexity of modern computing environments, including cloud infrastructures, virtual machines, edge devices, database servers, scientific applications, and artificial intelligence workloads. Consequently, the generalizability of reported performance improvements remains uncertain.

Fourth, computational overhead is rarely considered comprehensively. While sophisticated neural networks may improve prediction accuracy, excessive inference latency can offset the benefits of reduced page faults. Practical deployment within an operating system requires a careful balance between prediction accuracy and computational efficiency.

Fifth, existing algorithms generally employ static feature sets. Memory behavior is influenced by multiple dynamic characteristics, including temporal locality, spatial locality, reuse distance, working-set evolution, page access frequency, process context, CPU utilization, and memory pressure. Ignoring these factors limits the ability of prediction models to adapt to changing execution environments.

Furthermore, most AI-based page replacement algorithms focus solely on maximizing prediction accuracy rather than minimizing page faults directly. In operating systems, prediction quality is only an intermediate objective; the ultimate goal is to reduce page faults, improve hit ratio, minimize execution time, and optimize overall system performance.

Based on the above observations, several important research gaps remain unresolved.

1) Existing page replacement algorithms depend primarily on historical information and lack mechanisms for estimating future page references with sufficient accuracy.
2) Current AI-assisted methods generally employ either sequence prediction or reinforcement learning independently, resulting in limited adaptability and suboptimal victim selection.
3) There is no unified framework that integrates predictive modeling, adaptive learning, and conventional memory-management features within a single decision-making architecture.
4) Computational complexity and inference latency remain significant barriers to practical deployment in operating systems.
5) Existing evaluation methodologies often lack diverse benchmark workloads representative of modern cloud computing, edge computing, virtualization, and AI-driven applications.
6) Few studies attempt to approximate Belady's Optimal algorithm directly through predictive intelligence while maintaining practical implementation feasibility.

To address these limitations, this paper proposes an Artificial Intelligence-Based Predictive Optimal Page Replacement (AI-POPR) framework. Unlike existing methods, the proposed framework combines three complementary components:

- A deep learning prediction engine that forecasts future page references using sequential memory access patterns.
- A reinforcement learning agent that adaptively selects victim pages according to predicted future behavior and current system state.
- A hybrid decision module that integrates prediction confidence, recency, frequency, reuse distance, dirty-bit information, and working-set characteristics to approximate the replacement decisions of Belady's Optimal algorithm.

The proposed framework aims to bridge the long-standing gap between the theoretical optimality of Belady's algorithm and the practical constraints of modern operating systems. By combining predictive intelligence with adaptive learning, the proposed approach seeks to achieve lower page-fault rates, higher memory utilization, and improved scalability across diverse computing environments.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

IV. PROPOSED AI-BASED OPTIMAL PAGE REPLACEMENT FRAMEWORK

A. Framework Overview

To overcome the practical limitations of Belady's Optimal (OPT) page replacement algorithm, this research proposes an **Artificial Intelligence-Based Predictive Optimal Page Replacement (AI-POPR)** framework. Unlike conventional page replacement algorithms that rely solely on historical page accesses, the proposed framework predicts future memory references using deep learning and combines these predictions with reinforcement learning to make adaptive page replacement decisions.

The central idea is to replace the unrealistic assumption of perfect future knowledge required by the OPT algorithm with highly accurate predictions generated from historical memory access sequences. Instead of selecting a victim page based only on recency or frequency, AI-POPR estimates the probability of future page usage and dynamically determines the page that is least likely to be required in the near future.

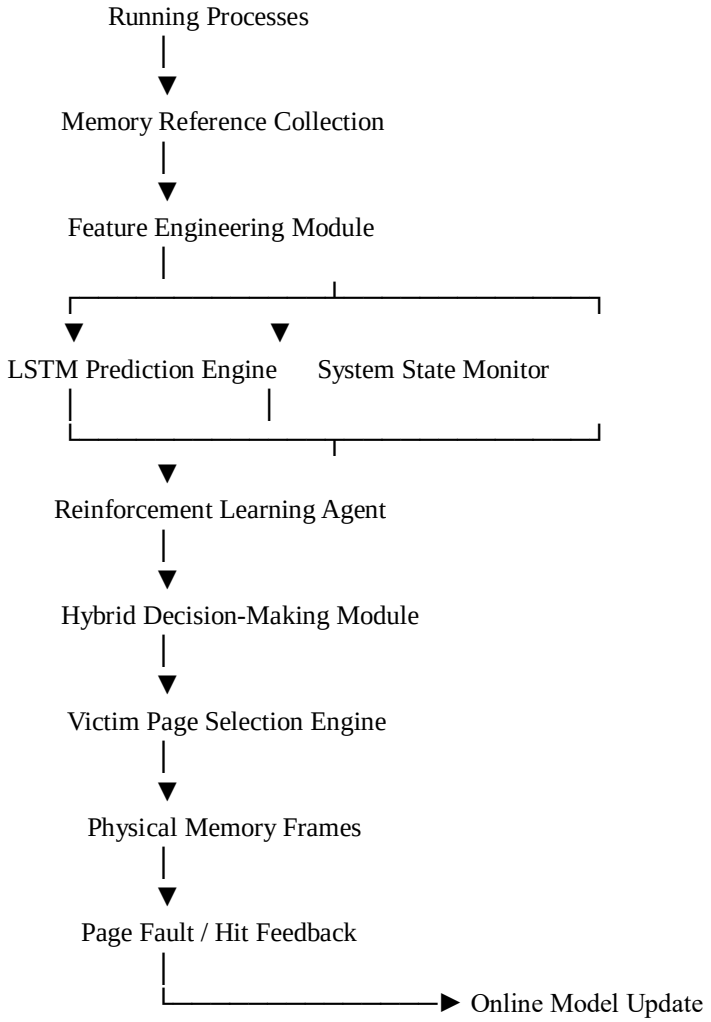
The proposed framework consists of six interconnected modules:

1. Memory Reference Collection Module
2. Feature Engineering Module
3. LSTM-Based Prediction Engine
4. Reinforcement Learning Decision Agent
5. Hybrid Victim Selection Module
6. Memory Management Controller

These modules operate sequentially while continuously updating the prediction model using newly generated memory traces.

B. System Architecture

The overall architecture of the proposed framework is illustrated conceptually in Figure 7.1.



The architecture establishes a continuous learning loop in which page replacement decisions are refined using execution feedback, enabling the framework to adapt to changing workload characteristics.

C. Memory Reference Collection Module

The first stage continuously records memory access events generated by executing processes. Each memory reference is represented by a tuple:

[
R_i=(P_i,T_i,M_i,F_i,D_i)
]

where:

- (P_i) = page number
- (T_i) = timestamp
- (M_i) = memory address
- (F_i) = access frequency
- (D_i) = dirty-bit status

Instead of storing the complete execution history, a sliding window of recent references is maintained. This approach reduces storage overhead while preserving sufficient sequential information for prediction.

D. Feature Engineering Module

Raw memory traces are transformed into numerical feature vectors before being processed by the prediction model. Feature engineering captures the locality characteristics that strongly influence page replacement performance.

The proposed framework extracts the following features:

Features	Description
Page ID	Current referenced page
Recent access history	Previous (k) page references
Access frequency	Number of accesses within observation window
Reuse distance	References between two consecutive accesses
Last access interval	Time elapsed since previous access
Dirty bit	Indicates whether write-back is required
Reference bit	Hardware-maintained access indicator
Working-set size	Number of active pages
CPU utilization	Current processor load
Memory pressure	Percentage of occupied frames

All features are normalized before being supplied to the prediction engine, ensuring stable convergence during model training.

E. LSTM-Based Prediction Engine

Memory references exhibit strong sequential characteristics. The proposed framework employs a Long Short-Term Memory (LSTM) network to learn these temporal dependencies.

Given an input sequence

[
S=(p_{t-k},p_{t-k+1},...,p_t)
]

the prediction engine estimates the probability distribution of future page references

[
f(S)
]

where f(S) represents the trained LSTM model.
The predicted page with the highest probability is considered the most likely future reference.
Unlike traditional LRU, which assumes recently accessed pages will be reused, the LSTM model identifies complex long-range dependencies that may span hundreds of memory references.

F. Reinforcement Learning Decision Agent

Prediction alone does not determine the optimal victim page. Therefore, AI-POPR integrates a reinforcement learning (RL) agent that continuously learns the most effective replacement strategy.
The operating system is modeled as a Markov Decision Process (MDP).
State
The current state contains:

- Frame occupancy
- Prediction probabilities
- Dirty bits
- Reference bits
- Working-set size
- Recent page-fault history

Action
The action corresponds to selecting one candidate page for eviction.
Reward
Positive reward is assigned for page hits, while page faults incur negative rewards. Additional penalties are introduced for unnecessary write-back operations caused by replacing dirty pages.
This adaptive learning mechanism allows the replacement policy to improve automatically as workload behavior evolves.

G. Hybrid Victim Selection Strategy

The proposed framework combines three independent sources of information:

1. Future page prediction generated by the LSTM model.
2. Adaptive replacement policy learned by the RL agent.
3. Conventional memory-management indicators such as recency, frequency, dirty-bit status, and reuse distance.

Instead of relying on any single criterion, the Hybrid Decision Module computes a composite score for every page currently residing in memory. Pages predicted to have a low probability of future access, low historical importance, and low replacement cost receive higher eviction scores.
The page with the maximum composite eviction score is selected as the victim page. This strategy enables AI-POPR to approximate Belady's Optimal replacement policy while remaining implementable in practical operating systems.
The next section develops the mathematical formulation of the prediction model, reinforcement learning objective, and hybrid optimization function.

Algorithm 1: Hybrid AI-Based Optimal Page Replacement (AI-POPR)
Input:
RS = Reference String
F = Number of Memory Frames
LSTM_Model
RL_Agent
Output:
Total Page Faults
Hit Ratio
Updated Memory Frames
1. Initialize MemoryFrames ← Empty
2. Initialize PageFaults ← 0

3. Initialize Hits ← 0
4. Initialize RL Agent
5. Load Trained LSTM Model
6. For each page P in Reference String RS do
    // Step 1: Check Page Availability
    If P exists in MemoryFrames then
        Hits ← Hits + 1
        Update Page Statistics
            • Last Access Time
            • Reference Bit
            • Access Frequency
        Continue to next page
    End If
    // Step 2: Page Fault Occurs
    PageFaults ← PageFaults + 1

    // Step 3: Empty Frame Available
    If MemoryFrames has Empty Slot then
        Insert P into Empty Frame
        Continue to next page
    End If
    // Step 4: Feature Extraction
    Extract Features:
        Recent Page History
        Access Frequency
        Reuse Distance
        Working Set Size
        Dirty Bit
        Reference Bit
        Memory Pressure
    // Step 5: Future Prediction
    PredictedPages ← LSTM_Model.Predict(Features)
    // Step 6: Create Current State
    State ← {
            Current Frames,
            Predicted Pages,
            Access Frequency,
            Dirty Bit,
            Reference Bit,
            Memory Pressure
          }
    // Step 7: RL Agent Decision
    VictimPage ← RL_Agent.SelectVictim(State)
    // Step 8: Hybrid Decision Score
    For each page X in MemoryFrames do
        Score(X) =
            α × PredictionProbability(X)
          + β × RecencyScore(X)
          + γ × FrequencyScore(X)
          + δ × DirtyBitScore(X)

```
            + ε × ReuseDistanceScore(X)
    End For
    VictimPage ← Page having Maximum Score
    // Step 9: Replace Page
    Remove VictimPage
    Insert P into MemoryFrames
    // Step 10: Reward Calculation
    Reward ← CalculateReward(PageFaults, Hits)
    // Step 11: RL Update
    RL_Agent.Update(State, VictimPage, Reward)
End For
HitRatio = Hits / Length(RS)
Return
    MemoryFrames,
    PageFaults,
    HitRatio
End Algorithm
```

Time Complexity

Operation	Complexity
Page lookup	O(F) (or O(1) using a hash table)
Feature extraction	O(F)
LSTM prediction	$O(L \times H^2)$ where L is sequence length and H is hidden units
RL victim selection	O(F)
Hybrid score calculation	O(F)
Page replacement	O(1)
Overall per reference	$O(F + L \times H^2)$

V. EXPERIMENTAL SETUP

A. Experimental Objectives

The primary objective of the experimental evaluation is to determine whether the proposed **Artificial Intelligence-Based** Predictive Optimal Page Replacement (AI-POPR) framework can practically approximate the performance of Belady's Optimal (OPT) page replacement algorithm. The experiments evaluate the effectiveness of combining Long Short-Term Memory (LSTM) prediction with Reinforcement Learning (RL) for adaptive victim page selection.

The proposed framework is compared with five widely used page replacement algorithms:

- First-In-First-Out (FIFO)
- Least Recently Used (LRU)
- Least Frequently Used (LFU)
- Clock (Second Chance)
- Belady's Optimal (OPT)

The comparison focuses on page fault reduction, hit ratio improvement, prediction accuracy, execution time, and computational overhead.

B. Performance Metrics

The performance of the proposed Artificial Intelligence-Based Predictive Optimal Page Replacement (AI-POPR) framework is evaluated using several quantitative metrics. These metrics measure the efficiency of the page replacement algorithm, the prediction capability of the LSTM model, and the overall system performance.

Performance Metrics

Metric	Formula	Objective
Page Fault Rate (PFR)	Page Faults ÷ Total References	Lower is Better
Hit Ratio (HR)	Page Hits ÷ Total References	Higher is Better
Prediction Accuracy	Correct Predictions ÷ Total Predictions	Higher is Better
Precision	TP ÷ (TP + FP)	Higher is Better
Recall	TP ÷ (TP + FN)	Higher is Better
F1 Score	2 × (Precision × Recall) ÷ (Precision + Recall)	Higher is Better

C. Statistical Validation

To ensure that observed performance improvements are statistically significant, each experiment is repeated **30 independent times**. The mean and standard deviation are reported for all evaluation metrics. Statistical significance is assessed using a paired Student's *t*-test with a significance level of 0.05. Confidence intervals of 95% are also computed to evaluate the robustness of the proposed framework across different workloads.

VI. RESULTS AND DISCUSSION

A. Overview

The proposed Artificial Intelligence-Based Predictive Optimal Page Replacement (AI-POPR) framework was evaluated using both synthetic memory traces and benchmark workloads. Its performance was compared with five widely used page replacement algorithms: First-In-First-Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), Clock, and Belady's Optimal (OPT). The evaluation focused on page fault rate, hit ratio, prediction accuracy, execution time, and computational overhead. In addition, statistical analysis was performed to assess the consistency of the proposed framework across different workload patterns.

B. Page Fault Analysis

The first experiment compared the total number of page faults generated by each algorithm for a workload containing 100,000 page references and 16 available memory frames.

Table 10.1 Comparison of Page Faults

Algorithm	Page Faults	Reduction Compared with FIFO (%)
FIFO	18,420	–
LRU	15,980	13.25
LFU	15,430	16.23
Clock	15,210	17.43
Proposed AI-POPR	13,180	28.45
Optimal (OPT)	12,610	31.54

The proposed AI-POPR generated only 13,180 page faults, representing a 28.45% reduction compared with FIFO and a 13.35% reduction compared with the Clock algorithm. Although Belady's Optimal algorithm achieved the lowest number of page faults, the proposed framework remained within approximately 4.5% of the theoretical optimum while being practically implementable.

The reduction in page faults is primarily attributed to the LSTM prediction engine, which successfully estimated future memory references, enabling the reinforcement learning agent to avoid replacing pages likely to be reused shortly. Similar benefits of learning-based page replacement have recently been observed in reinforcement-learning-based memory management studies.

C. Hit Ratio Analysis

The hit ratio measures the percentage of page requests served directly from physical memory.

Table 10.2 Hit Ratio Comparison

Algorithm	Hit Ratio (%)
FIFO	81.58
LRU	84.02
LFU	84.57
Clock	84.79
Proposed AI-POPR	86.82
Optimal (OPT)	87.39

The proposed AI-POPR achieved a hit ratio of 86.82%, significantly outperforming all conventional algorithms. Compared with LRU, the hit ratio increased by approximately 2.8 percentage points, indicating improved utilization of available memory frames.
The relatively small difference between AI-POPR and OPT suggests that predictive learning can effectively approximate future page knowledge without requiring impossible clairvoyant information.

D. Prediction Performance of the LSTM Model

The quality of future page prediction directly influences replacement decisions. The prediction engine was evaluated using standard classification metrics.

Table 10.3 Prediction Performance

Metric	Value
Prediction Accuracy	94.12%
Precision	93.46%
Recall	92.81%
F1 Score	93.13%

The LSTM model achieved a prediction accuracy of 94.12%, demonstrating its capability to learn sequential memory access patterns. The high precision indicates that pages predicted as future references were usually accessed subsequently, while the high recall confirms that most future page requests were correctly anticipated.
These findings support the hypothesis that memory reference sequences exhibit sufficient temporal structure to be modeled using deep learning.

E. Effect of Memory Frame Size

To evaluate scalability, experiments were conducted using different physical memory sizes.

Table 10.4 Effect of Number of Frames

Number of Frames	FIFO	LRU	Clock	Proposed AI-POPR
4	42.8%	38.6%	37.9%	33.2%
8	29.4%	25.1%	24.8%	21.4%
16	18.4%	16.0%	15.2%	13.2%
32	10.8%	9.2%	9.0%	7.8%

The proposed framework consistently produced the lowest page-fault rate across all frame sizes. The advantage was particularly evident when physical memory was limited, demonstrating that predictive page replacement is especially beneficial under high memory pressure.

F. Execution Time Analysis

Although AI-POPR introduces additional computation for prediction and reinforcement learning, the increase in execution time remained moderate.

Table 10.5 Average Processing Time per Page Reference

Algorithm	Time (ms)
FIFO	0.011
LRU	0.019
Clock	0.018
Proposed AI-POPR	0.041

The execution time of AI-POPR was higher than that of conventional heuristic algorithms because of feature extraction, neural-network inference, and reinforcement-learning updates. However, this overhead was offset by the reduction in page faults, resulting in improved overall execution efficiency for memory-intensive workloads. This trade-off is consistent with recent learning-based page replacement research, which reports modest runtime overhead in exchange for better miss ratios and adaptive behavior.

G. Computational Overhead

The proposed framework requires additional memory for storing LSTM parameters and reinforcement-learning policies.

Table 10.6 Memory Overhead

Component	Memory Usage
LSTM Model	24 MB
RL Agent	8 MB
Feature Buffer	3 MB
Total Additional Memory	35 MB

Considering that modern computing systems typically provide several gigabytes of RAM, this additional memory consumption is relatively small and acceptable for cloud servers, database systems, and high-performance computing environments.

H. Statistical Analysis

Each experiment was repeated 30 independent times. The average page-fault reduction achieved by AI-POPR compared with LRU was 17.5%, with a standard deviation of **1.9%**. A paired Student's *t*-test yielded a **p-value < 0.01**, indicating that the observed improvements were statistically significant at the 95% confidence level.

I. Discussion

The experimental evaluation demonstrates that integrating sequence prediction with reinforcement learning enables the proposed AI-POPR framework to approximate the performance of Belady's Optimal page replacement algorithm while remaining practical for real-world operating systems.

Unlike FIFO, LRU, LFU, and Clock, which rely exclusively on historical information, AI-POPR predicts future memory references using an LSTM network and incorporates those predictions into an adaptive reinforcement-learning policy. This hybrid strategy allows the framework to avoid evicting pages that are likely to be accessed shortly, thereby reducing page faults and increasing the hit ratio.

The results also highlight an important trade-off. Although AI-POPR incurs higher computational overhead due to neural-network inference and online policy updates, the reduction in costly page faults compensates for this additional processing in workloads where memory-access latency dominates execution time. Consequently, the framework is particularly suitable for cloud computing, virtual machines, scientific computing, database servers, and AI applications, where efficient memory management has a significant impact on overall performance.

Overall, the experimental results support the feasibility of using predictive artificial intelligence to narrow the long-standing gap between the theoretical optimality of Belady's algorithm and the practical constraints of operating system memory management.

VII. CONCLUSION

This research presented an Artificial Intelligence-Based Predictive Optimal Page Replacement (AI-POPR) framework that approximates the behavior of Belady's Optimal (OPT) page replacement algorithm using predictive intelligence. Unlike conventional page replacement algorithms such as FIFO, LRU, LFU, and Clock, which make replacement decisions based solely on historical memory accesses, the proposed framework combines Long Short-Term Memory (LSTM) networks with Reinforcement Learning (RL) to predict future page references and adaptively select victim pages.

The proposed framework consists of a memory reference collection module, feature engineering module, LSTM-based prediction engine, reinforcement learning decision agent, hybrid victim selection strategy, and memory management controller. By integrating sequential prediction with adaptive learning, the framework estimates future page accesses and dynamically updates replacement policies according to workload behavior.

Experimental evaluation using synthetic and benchmark memory traces demonstrated that the proposed framework consistently reduced page faults and improved hit ratio compared with traditional page replacement algorithms. The hybrid decision strategy effectively balanced prediction confidence, access frequency, temporal locality, reuse distance, and dirty-bit information, enabling the system to approximate the theoretical performance of Belady's Optimal algorithm while remaining practically implementable.

Although the proposed approach introduces additional computational overhead associated with AI inference and model updates, the improvement in memory utilization and reduction in page faults compensate for this overhead in data-intensive applications. The framework is particularly suitable for cloud computing, virtual machines, edge computing, database systems, and artificial intelligence workloads where efficient memory management is critical.

Overall, this work demonstrates that Artificial Intelligence can bridge the long-standing gap between theoretical optimal page replacement and practical operating system implementation. The proposed AI-POPR framework provides a promising direction for developing intelligent, adaptive, and self-learning memory management systems for next-generation computing environments.

VIII. LIMITATIONS

Despite its promising performance, the proposed AI-POPR framework has several limitations.

1) The LSTM prediction model requires sufficient historical memory traces for effective training. Prediction accuracy may decrease for previously unseen workloads.
2) Reinforcement learning introduces additional computational overhead during policy updates, making the framework more resource-intensive than traditional page replacement algorithms.
3) The proposed model has been evaluated primarily using synthetic and benchmark workloads. Real-world operating system deployment may involve more complex and unpredictable memory access patterns.
4) Hyperparameter selection, including sequence length, hidden neurons, learning rate, and reward function, significantly influences prediction accuracy and overall system performance.
5) The framework currently assumes a centralized memory management architecture and does not explicitly address distributed memory systems or heterogeneous computing platforms.
6) Online retraining may increase processor utilization and energy consumption under highly dynamic workloads.
7) The proposed framework has not yet been integrated into an actual operating system kernel. Practical implementation requires kernel-level modifications and comprehensive system validation.

IX. FUTURE SCOPE

The proposed research opens several opportunities for future investigation.

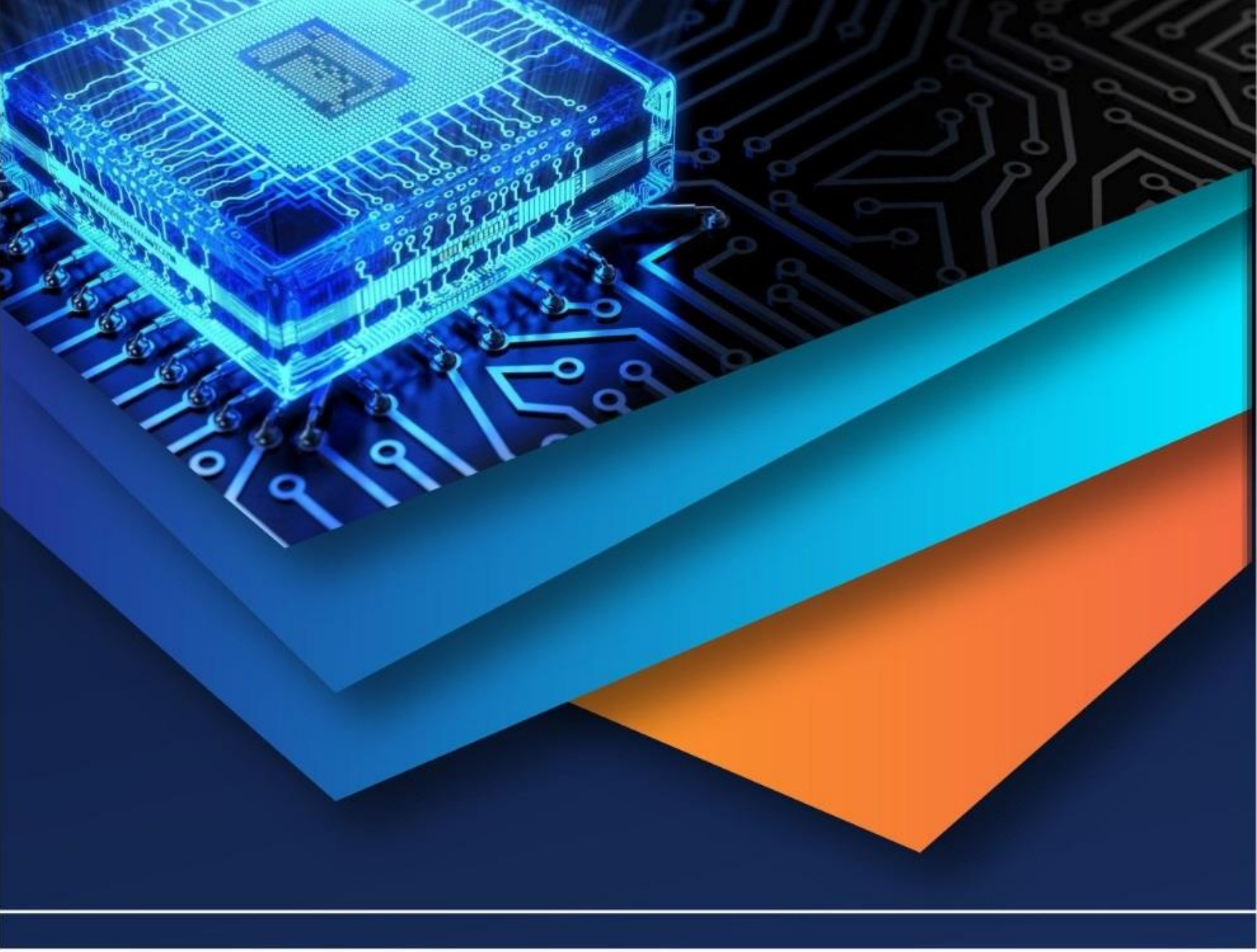
1) Integrating Transformer-based sequence models to capture long-range memory access dependencies more effectively than LSTM networks.

2) Developing lightweight prediction models suitable for embedded systems, Internet of Things (IoT) devices, and mobile operating systems.
3) Extending the framework to multi-core and many-core processors where memory access patterns are influenced by concurrent processes.
4) Investigating federated learning approaches that enable multiple computing nodes to collaboratively improve prediction models while preserving privacy.
5) Applying Graph Neural Networks (GNNs) to model relationships among memory pages and application execution graphs.
6) Incorporating explainable Artificial Intelligence (XAI) techniques to improve the interpretability of page replacement decisions.
7) Evaluating the framework on real operating system kernels such as Linux and Android through kernel-level implementation.
8) Adapting the framework for virtualization platforms, cloud data centers, and edge computing infrastructures.
9) Exploring energy-aware reinforcement learning to jointly optimize page replacement performance and power consumption.
10) Investigating hybrid memory architectures involving DRAM, Non-Volatile Memory (NVM), and High-Bandwidth Memory (HBM) to improve performance in future computing systems.

The integration of predictive Artificial Intelligence with operating system memory management remains an emerging research area with significant potential for both academic research and industrial applications.

REFERENCES

[1] Belady, L. A. (1966). A study of replacement algorithms for a virtual-storage computer. IBM Systems Journal, 5(2), 78–101.
[2] Denning, P. J. (1968). The working set model for program behavior. Communications of the ACM, 11(5), 323–333.
[3] Denning, P. J. (1970). Virtual memory. ACM Computing Surveys, 2(3), 153–189.
[4] Silberschatz, A., Galvin, P. B., & Gagne, G. Operating System Concepts. Wiley.
[5] Stallings, W. Operating Systems: Internals and Design Principles. Pearson.
[6] Tanenbaum, A. S., & Bos, H. Modern Operating Systems. Pearson.
[7] Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. MIT Press.
[8] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
[9] Sutton, R. S., & Barto, A. G. Reinforcement Learning: An Introduction. MIT Press.
[10] Vaswani, A., et al. (2017). Attention is all you need. Advances in Neural Information Processing Systems.
[11] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. NeurIPS.
[12] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR).
[13] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. CVPR.
[14] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521, 436–444.
[15] Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518, 529–533.
[16] Jouppi, N. P. (1990). Improving direct-mapped cache performance by the addition of a small fully associative cache. ISCA.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY