People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

IBN KHALDOUN UNIVERSITY OF TIARET

Dissertation

Presented to:

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTEMENT OF COMPUTER SCIENCE

in order to obtain the degree of :

MASTER

Specialty:
Artificial Intelligence and Digitalization
Presented by:

BENALI NOUR EL HOUDA

On the theme:

Towards intelligent caching in NDN networks

Defended publicly on 16 / 06 /2025 in Tiaret in front the jury composed of: :

Mr Alem Abdelkader	grade	Tiaret University	Chairman
Mr Nassane Samir	grade	Tiaret University	Supervisor
Mr Benouda Habib	grade	Tiaret University	Examiner
Mr. Sid Ahmed Mokhtar Mostefaoui	grade	Tiaret University	Co-Supervisor

2024-2025

Acknowledgment

First and foremost, we would like to thank Allah for given us the patience, the courage, the strength and health to complete this projec . To My supervisor, Mr Nassane Samir for all the time he has devoted to Me, for his valuable advice and for all his help and support during the realization of this project. To Mr Sid Ahmed Mokhtar Mostefaoui, for all the add he give it to me for the realization of this project. Finally, we also want to thank the members of the jury for accepting to evaluate our work.

Dedication

To the little me, to the grown me
Thank you

Table of Contents

Chapter-II- AI-based cache replacement polices

Liste des figures

Abstract

Can we develop intelligent strategies specific to NDN networks that outperform traditional approaches and optimize the performance of these NDN networks? This question guided the course of this dissertation, driven by the inherent limits of the present IP-based Internet paradigm and the rising shift toward data-centric architectures. The significance of this question derives from the crucial role caching plays in improving latency, bandwidth utilization, and scalability in NDN, and the inability of traditional caching techniques to adapt to dynamic user behavior.

This research tested and confirmed several hypotheses: (1) reinforcement learning methods can dynamically outperform fixed cache replacement strategies, and (2) combining spatial and temporal learning components—specifically CNNs and LSTMs—improves the decision-making capability of RL-based caching models.

To address these hypotheses, A novel architecture for model free-reinforcement learning was proposed based on Dueling DQN, integrating CNN with LSTM that enables spatial pattern extraction from content request distributions, while LSTM captures temporal dependencies of request trends over time.

This development of an intelligent caching framework led this dissertation, required for the Master's degree in Computer Science, proved significant theoretical improvements in responsiveness and adaptability to shifting request distributions. These findings provide credence to the idea that optimizing cache replacement in NDN systems may be achieved by deep reinforcement learning.

This work remains focused on the fundamental research question throughout, providing a clear, concise roadmap that represents the logic and depth of the study.

Key words:

Named data networking (NDN), caching, intelligent caching replacement policies, Deep reinforcement learning (DRL).

ملخص

هل يُمكننا تطوير استراتيجيات ذكية خاصة بشبكات NDN تتفوق على الأساليب التقليدية وتُحسّن أداء هذه الشبكات؟ وجّه هذا السؤال مسار هذه الأطروحة، مدفوعةً بالقيود المتأصلة في نموذج الإنترنت الحالي القائم على بروتوكول الإنترنت (IP) والتحول المتزايد نحو البنى المُركّزة على البيانات. تنبع أهمية هذا السؤال من الدور الحاسم الذي يلعبه التخزين المؤقت في تحسين زمن الوصول، واستخدام النطاق الترددي، وقابلية التوسع في شبكات NDN، وعدم قدرة تقنيات التخزين المؤقت التقليدية على التكيف مع سلوك المستخدم الديناميكي.

اختبر هذا البحث وأكد عدة فرضيات: (1) يمكن لأساليب التعلم المُعزّز أن تتفوق ديناميكيًا على استراتيجيات استبدال ذاكرة التخزين المؤقت الثابتة، و(2) يُحسّن الجمع بين مكونات التعلم المكاني والزماني - وتحديدًا شبكات CNN ووحدات LSTM - قدرة نماذج التخزين المؤقت القائمة على التعلم المُعزّز على اتخاذ القرارات.

لمعالجة هذه الفرضيات، اقتُرحت بنية جديدة لنموذج التعلم التعزيزي الحر، استنادًا إلى Dueling DQN، تدمج CNN مع LSTM، مما يُمكّن من استخراج الأنماط المكانية من توزيعات طلبات المحتوى، بينما تلتقط LSTM التبعيات الزمنية لاتجاهات الطلب بمرور الوقت.

أدى هذا التطوير لإطار عمل ذكي للتخزين المؤقت إلى إثبات هذه الأطروحة، المطلوبة لنيل درجة الماجستير في علوم الحاسوب، تحسينات نظرية مهمة في الاستجابة والقدرة على التكيف مع توزيعات الطلبات المتغيرة. تُعزز هذه النتائج فكرة إمكانية تحقيق تحسين استبدال ذاكرة التخزين المؤقت في أنظمة NDN من خلال التعلم التعزيزي العميق.

يظل هذا العمل مُركزًا على سؤال البحث الأساسي، مُقدمًا خارطة طريق واضحة وموجزة تُمثل منطق الدراسة وعمقها.

الكلمات المفتاحية:

شبكات البيانات المسماة (NDN)، التخزين المؤقت، سياسات استبدال التخزين المؤقت الذكي، التعلم التعزيزي العميق (DRL).

Résumé

Peut-on développer des stratégies intelligentes spécifiques aux réseaux NDN qui surpassent les approches traditionnelles et optimisent la performance de ces réseaux NDN?Cette question a guidé cette mémoire , motivée par les limites inhérentes au paradigme actuel de l'Internet basé sur IP et la transition croissante vers des architectures centrées sur les données. L'importance de cette question découle du rôle crucial de la mise en cache dans l'amélioration de la latence, de l'utilisation de la bande passante et de l'évolutivité des réseaux NDN, ainsi que de l'incapacité des techniques de mise en cache traditionnelles à s'adapter au comportement dynamique des utilisateurs.

Cette recherche a testé et confirmé plusieurs hypothèses : (1) les méthodes d'apprentissage par renforcement peuvent surpasser dynamiquement les stratégies de remplacement de cache fixe ; et (2) la combinaison de composants d'apprentissage spatial et temporel, notamment les CNN et les LSTM, améliore la capacité de prise de décision des modèles de mise en cache basés sur l'apprentissage par renforcement.

Pour répondre à ces hypothèses, une nouvelle architecture d'apprentissage par renforcement sans modèle a été proposée, basée sur le Dueling DQN. Elle intègre les CNN et les LSTM, permettant ainsi l'extraction de modèles spatiaux à partir de la distribution des requêtes de contenu, tandis que les LSTM capturent les dépendances temporelles des tendances des requêtes au fil du temps.

Le développement d'un framework de mise en cache intelligent a conduit à cette mémoire, requise pour le master en informatique, qui a démontré des améliorations théoriques significatives en termes de réactivité et d'adaptabilité aux variations de distribution des requêtes. Ces résultats confortent l'idée que l'optimisation du remplacement du cache dans les systèmes NDN peut être obtenue par apprentissage par renforcement profond.

Ce travail reste centré sur la question de recherche fondamentale, fournissant une feuille de route claire et concise qui illustre la logique et la profondeur de l'étude.

Mots clés :

Réseaux de données nommés (NDN), mise en cache, politiques de remplacement du cache intelligent, apprentissage par renforcement profond (DRL).

General Introduction

General Introduction

The exponential growth in data traffic alongside the rapid increase in consuming content such as in social networking platforms, and the exploded use of streaming in recent years, has revealed the inadequacies of the existing IP-based Internet structure, particularly regarding its mobility, scalability, and content delivery efficiency. A proposed data-centric paradigm, Named Data Networking (NDN)) emerged as promising solution, attempts to fix some of these issues, given its distributed content caching system, where data can be cached in multiple routers and retrieved from the closest one instead of the original producer, enhancing content availability, reducing latency, and minimizing data loss.

Nevertheless, NDN's performance relies heavily on its caching policies, particularly on the replacement strategies employed at the fill-up point. The traditional approaches like Least Recently Used (LRU), Least Frequently Used (LFU), and their updates are based on heuristic rules, which most of times fail to adapt to the dynamic nature of network traffic. This results in inefficient cache usage and increased response latency.

To address these challenges, recent research has focused on intelligent caching solutions that use machine learning, notably reinforcement learning (RL). Instead of depending on static rules, these approaches enable cache replacement decisions to be learnt and optimized over time based on observed network behavior. This shift allows for a more adaptable and context-aware caching techniques that respond to swings in content popularity, request patterns, and network circumstances.

This dissertation explores the transition from traditional caching strategies to intelligent learning based approaches in the context of NDN, starting by outlining the underlying limits of the IP-based Internet and the NDN architecture, Subsequently, formulate the cache replacement problem as a Marcov decision process (MDP), An intelligent agent may dynamically adapt its behavior based on the changing network state and request patterns, and explores a set of progressively sophisticated models, starting from Q-Learning and progressing to Deep Q-Networks (DQN) and Double DQN (DDQN) highlighting their relevance and efficiency

The final part of the study propose a novel intelligent caching policy based on the dueling DQN, where it integrates LSTM and CNN, to further capture spatial and temporal correlation in content request patterns.

Our results show that this architecture outperforms conventional strategies in terms of hit ratio, reduced content retrieval latency, and adaptability to dynamic workloads. The integration of CNN enables spatial pattern extraction from content request distributions, while LSTM captures temporal dependencies of request trends over time. Enhanced decision-making by the Dueling DQN agent improves context-sensitive cache replacement decision precision.

1.1 Background

The U.S. National Science Foundation's Future Internet Architecture Program funds five research initiatives, including Named Data Networking (NDN) [2]. It is a paradigm shift from old host-centric communication models data-centric Internet architecture [3], which marks a substantial transformation in the way networks operate. NDN modifies the semantics of the network service so that it retrieves data identified by a provided name instead of delivering the packet to a specified destination. This seemingly simple adjustment has far-reaching consequences for how we design, develop, deploy, and utilize networks and apps.
NDN's significance stems from its capacity to address issues with scalability, aids in bandwidth reduction, eases network congestion [4], lower latency, and improve security by protecting data directly rather than access points. NDN incorporates content naming and retrieval directly into its design, in contrast to IP networks that need middleware to translate application-specific models to network delivery techniques. This removes the inefficiencies that come with conventional routing and promotes reliable communication in a variety of settings, such as mobile networks and the Internet of Things.

The foundational ideas of NDN's architecture are named data packets, stateful forwarding planes, and hop-by-hop flow balance. These capabilities enable routers to remove unnecessary data exchanges, effectively handle upcoming requests, and store material locally. By substituting hierarchical data names for IP addresses, NDN's thin waist allows for smooth scaling and mobility support while preserving interoperability with current Internet infrastructure.

NDN allows routers to cache data packets at multiple points in the network. This distribution caching mechanism not only improves data access speed [5], but also allows future requests for the same content to be served directly from intermediate nodes rather than the original source [6].

Web caching relies heavily on the cache replacement policy. The high level of sophistication in the cache system requires these replacement methods. By removing the item from the cache and creating room for the new object, these replacement rules are useful. A cache cannot hold the complete requested item due to its limited size. Consequently, we make space for new documents by using the cache replacement policy. This is relevant when there are already too many objects in the cache and we need to add more. To create space, we must remove the item from the cache. When it comes to the web cache, several cache replacement rules are crucial [7].

However the intrinsic restrictions of traditional cache replacement strategies, such Least Recently Used (LRU) and Least Frequently Used (LFU), make it difficult for them to make the best caching choices in dynamic network situations. These baseline policies frequently have trouble adjusting to changing user request patterns, shifting content popularity, and shifting network conditions, which leads to less than ideal cache usage and decreased performance. As a result, these inefficiencies raise network traffic, decrease cache hit rates, and increase latency.

In order to overcome these obstacles, intelligent and flexible caching techniques that can make defensible choices are desperately needed. By utilizing sophisticated methods like reinforcement learning, particularly an advanced architecture of dueling deep Quality network (duel DQN), caching policies can dynamically adapt to changing network conditions and content demands.

Named Data Networking (NDN) efficiency might be greatly increased by such smart rules through better cache management, faster content delivery, lower latency, eventually, these developments would allow NDN networks to respond more effectively to user needs while guaranteeing scalability and higher performance in a variety of situations.

1.2 Research problem

1.2.1 Problem identification: Named Data Networking (NDN) caching techniques now in use suffer from serious inefficiencies, such as higher latency and lower cache hit rates. Traditional policies like Least Recently Used (LRU) and Least Frequently Used (LFU) are unable to dynamically adjust to changing network circumstances, content popularity, and user request patterns, which leads to these problems. The user experience and network performance are adversely affected by the ensuing inefficient caching choices.

1.2.2 Problem statement: How can NDN networks' caching decisions be improved using deep reinforcement learning approaches to increase cache hit rates, lower latency, and improve network performance overall? This research specifically attempts to investigate the

use of sophisticated reinforcement learning models, including proposed model based on dueling DQ-Networks (Duel DQN), to intelligently modify caching strategies in dynamic NDN settings.

1.3 Objectives

This study aims to overcome the drawbacks of conventional caching policies by presenting a deep reinforcement learning strategy based on prediction that can make intelligent caching choices in complicated and dynamic NDN settings. In order to verify the influence of the suggested solution on important performance measures including cache hit ratio, latency, server load, and overall network efficiency. The main objective is to build a Deep Reinforcement Learning (DRL) model to create and assess an intelligent cache replacement policy for Named Data Networking (NDN). In order to maximize performance on NDN networks, increase the effectiveness of replacement decisions and optimize cache resource use. This strategy seeks to dynamically modify caching decisions based on network circumstances and content popularity.

1.4 Methodology

This research adopts a structured analytical and design-based methodology. It begins with a critical review of traditional caching replacement strategies used in Named Data Networking (NDN), highlighting their limitations in dynamic and data-intensive environments. The study then explores intelligent caching approaches grounded in reinforcement learning, particularly focusing on Q-Learning, Deep Q-Networks (DQN), and Double DQN. Building on these foundations, a novel architecture is proposed that integrates the Dueling DQN framework with Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) layers. This hybrid model aims to enhance decision-making at cache replacement points by capturing both spatial and temporal patterns in network traffic.

The suggested solution is then compared to baseline techniques using key performance characteristics such as cache hit ratio, average latency, and network traffic load to determine its efficiency and adaptability.

1.5 Structure of the dissertation

My dissertation is organized into 3 chapters:

General introduction: This chapter provides an overview of the context, objectives, methodology, and structure of this research work, setting the stage for a full examination of improving caching decisions in Named Data Networking (NDN) with deep reinforcement learning.

Chapter one: In this chapter, I look at the beginnings of Named Data Networking (NDN), tracing its roots back to Information-Centric Networking (ICN) and explaining why NDN has become such an important solution for overcoming the limits of traditional IP-based systems. Additionally, this research emphasizes NDN's architectural principles and benefits over traditional networking architectures.

Chapter two: In his chapter I provide an in-depth introduction to machine learning going through its types, focusing on the concept and start of reinforcement learning. It explores how Markov Decision Processes (MDPs) formalize caching problems. Additionally, it delves into intelligent cache replacement policies and outlines the approaches used to model cache replacement as a reinforcement learning problem

Chapter three: This chapter describes the implementation of intelligent caching in NDN utilizing advanced deep reinforcement learning, proposing an advanced model based on Dueling DQN. The Dueling architecture distinguishes between value and advantage functions to properly determine state importance. The updated approach use both (CNN) and (LSTM) to detect spatial and temporal patterns in request data to optimize cache decisions. Zipf-distributed requests and experience replay are used during training to ensure steady learning. Evaluation reveals that the suggested model outperforms traditional strategies and standard RL-based techniques in terms of cache hit ratio and latency.

Chapter 1

NAMED DATA NETWORKING

1. Introduction

The Internet has grown dramatically since its start in the 1960s, from a simple communication network to a dynamic platform that underlies modern life. Initially developed as a point-to-point communication system, the TCP/IP protocol stack allowed the delivery of text, audio, and video packets, establishing the groundwork for global connection. Due in large part to the exponential rise of user-generated content, content-centric services like YouTube, Netflix, Amazon, and social networking platforms, the purpose of the Internet has changed over time from sharing resources to distributing and retrieving vast amounts of information.[8][9]

The classic host-centric IP-based architecture is resilient, but it can't keep up with the demand of modern applications. The Internet does not come with strong mobility or security capabilities by default, nor was it built to accommodate content distribution models that value data above location. These limitations are handled via add-ons or patches, which frequently fail to scale properly.

Researchers have put up Information-Centric Networking (ICN) [10][11]as a ground-breaking method of Internet design in order to address these issues. ICN replaces host-centric communication with content-centric networking by using unique names instead of IP addresses to identify and route data. This makes it possible for location-independent data retrieval, many-to-many communication, and effective in-network caching. Among ICN designs, Named Data Networking (NDN) is notable for its capacity to handle large-scale information dissemination and revolutionize communication patterns.

2. The limits of the current Internet architecture

2.1. The internet's best effort delivery service paradigm does not guarantee speed for individual applications. Apps like email, online access, and file transfers have operated with this type of service, but modern apps like live audio and video streaming demand more than just the fastest possible performance.

2.2. Security : [12]

a) It is difficult to encrypt data that moves over the network by default since the existing design does not provide encryption by default. This implies that sensitive information can be intercepted and viewed by unauthorized third parties.

b) Protecting against cyber-attacks and cybercrimes is challenging due to the absence of integrated security methods for data transmission and routing.

Absence of access control: To safeguard network resources, the existing design lacks strong access control methods. Security problems including illegal access to servers and private information may result from this.

2.3 Content distribution:

a) Latency and congestion: The existing design may cause latency and congestion problems when content are distributed. It may take a while for data to get to its destination, particularly if it has pass through many routers.

b) Bandwidth inequality: An equal allocation of bandwidth between users and content producers is not guaranteed by the existing design. When streaming online, this might lead to performance problems, particularly when demand is strong.

c) Content distribution is heavily influenced by Internet service providers (ISPs). Unfair access to content may result from traffic management practices they put in place that prioritize some content over others.

d) Challenges for new content distributors: Due to the way the Internet is now set up, it may be challenging for new content distributors to compete with established providers. This may restrict how widely content is distributed online.

2.4 Scalability of routing: The existing BGP-based routing system is not built to manage the rapidly growing number of Internet-connected devices and connections. The routing system is under stress because to exponential expansion in data and traffic, which can cause outages and congestion. It is challenging to control data flows and maximize performance when routing is rigid and unadaptable [13].

2.5 Interoperability and Fragmentation: The network has been split up into several separate groups due to the absence of a common internet architecture, which has made it difficult for different services and applications to communicate with one another. This separation impedes innovation and the development of new services by making data transfers across systems more difficult. Therefore, the future of the internet depends on creating a more open and interoperable network [14].

2.6 Data centralization: Concentrating information on certain servers or data centers is known as data centralization. The basic client-server approach involves clients requesting data or services from centralized servers. However, scalability and data availability issues are brought on by this concentration.

2.7 Inefficient handover: It is possible for there to be delays and service disruptions while moving a mobile device between access points. This may result in problems with connections and disruptions to running apps.

3. Approaches for the Future Internet Architecture

A result of the above-mentioned issues and the drastic shift in Internet usage is the content-oriented network (CCN) strategy. Regardless of the hosts, the goal of this strategy is to distribute content. It incorporates caching natively and views named content as the network's central component. As a result, a copy of the requested content may be obtained from the network's most suitable node, satisfying the requirements for more effective content distribution than the existing Internet. Furthermore, mobility is no longer an issue because content names serve as identifiers separate from the locators. Lastly, by including cryptographic techniques into the content itself and employing a suitable naming scheme, the ICN substitutes a content-based model for the conventional concept of connection security in order to satisfy security requirements [15].

This section presents -Named data networking approaches to the future internet architecture

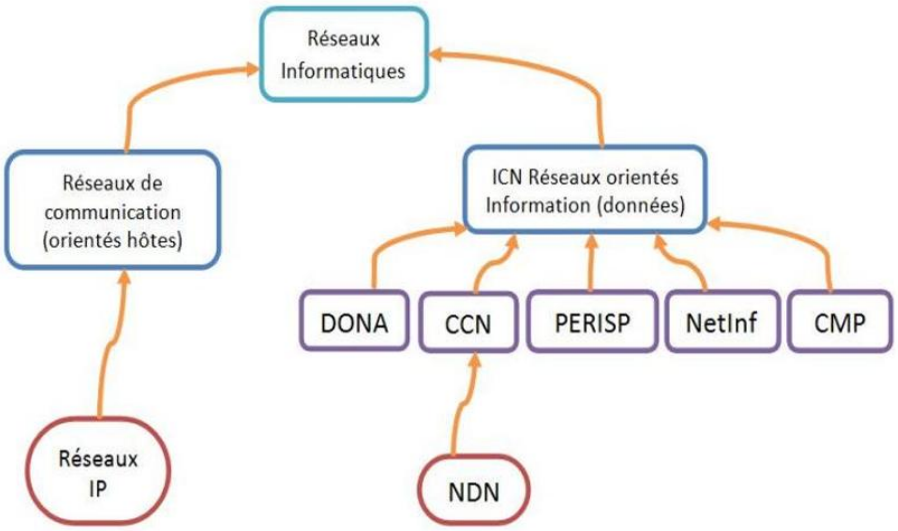

Figure 1 NDN and the main architecture [16]

4. Vision: A New Narrow Waist

The hourglass design of the modern Internet is based on a universal network layer, or IP, which provides the bare minimum of functionality required for worldwide interconnection. This thin waist facilitated the Internet's rapid expansion by allowing lower and top layer technologies to evolve independently. IP, on the other hand, was created to establish a

communication network in which packets were identified solely for communication endpoints. The Internet is now widely used as a distribution network due to the steady rise of e-commerce, digital media, social networking, and smartphone apps.

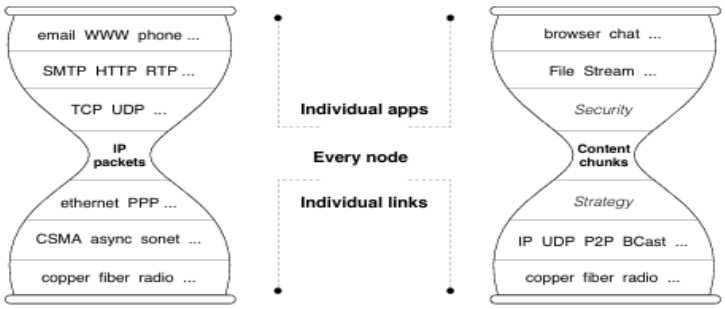

Figure 2 Internet and NDN Hourglass Architectures [17]

5. ICN and CCN : the origins of Named data Networking

5.1 Information centric network (ICN)

The late 2000s saw the emergence of the novel paradigm known as Information-Centric Networking (ICN), which shifted the emphasis from host-to-host communication to the retrieval of information objects by name in response to the increasing need for scalable and effective content distribution over the Internet. Through significant European and international research projects like 4WARD, NetInf, PSIRP, and DONA, key researchers like Bengt Ahlgren, Christian Dannewitz, Dirk Kutscher, and Börje Ohlman played crucial roles in shaping ICN. The IRTF's Information-Centric Networking Research Group (ICNRG) was formally chartered in 2010 and is led by David R. Oran and Dirk Kutscher [18]. For applications ranging from online content to IoT and mobile video, ICN systems use name-based routing, in-network caching, and replication to divorce content from its location and enable scalable, reliable, and effective data delivery.

Compared to traditional IP networks, ICN offers several benefits. First, by employing content caching at several network tiers, it makes it possible for more effective content delivery. This enhances overall performance and lessens network congestion. Furthermore, by using digital signature techniques to confirm the content's integrity and validity, the ICN offers improved security. But there are issues with the ICN as well. Considerations such as content name management and access control procedures are intricate. Deployment and compatibility work are also necessary when switching from ICN architecture to a conventional IP infrastructure. Although caching speeds up data distribution in information-

centric networks, cache placement and administration still present difficulties that need for more study.

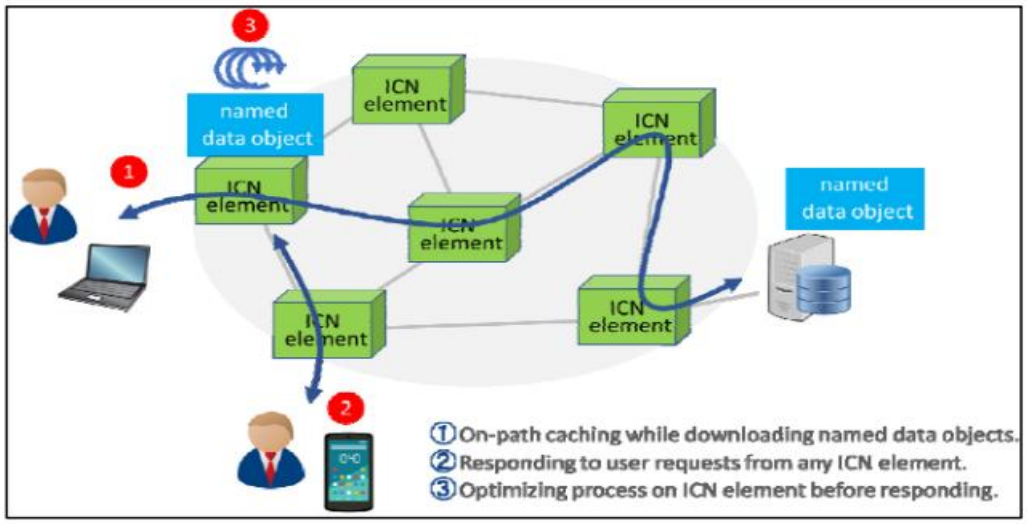

Figure 3 basic operation of ICN

5.2 Content centric network (CCN)

One of the most popular and significant ICN designs is Content-Centric Networking (CCN), which was first presented by Van Jacobson and his group at PARC in 2009. With the intention of resolving a number of issues with the Internet's present architecture, specifically those pertaining to data management, security, and device mobility [19].

The hourglass architecture of the existing internet is preserved in CCN, but names rather than IP addresses are used to store and retrieve the data. One of the two primary levels of this architecture is the "strategy" layer, which enables the control of network data flows (requests and answers). as well as a "Security" layer to ensure the authenticity, confidentiality, and integrity of the data sent between network nodes [20].

CCNx is the name of the CCN architecture's current implementation. Cisco created CCNx to expand the CCN concept by includes cutting-edge capabilities like Quality of Service (QoS) and mobility support. Additionally, it can help with the effective dissemination of data amongst linked devices in applications like the Internet of Things (IoT) [19] [21].

6. Named data networking :

6.1 Introduction :

The main NDN idea and architecture were outlined in the NDN project paper by L. Zhang et al [22].Which also claimed that "NDN is a universal Overlay" similar to IP. NDN is a receiver-based, data-centric communication protocol. In NDN, two distinct packet types are used for every communication. Both of them have names that identify the necessary data, and

we call them "Interest" and "Data." All that is required of the consumer is to include the name of the necessary material in an interest packet and send it over the network. The router forwards it to the data producer using the data name. The data whose name most closely matches the requested one is sent back to the customer once the names have been matched. Every data packet has a signature to firmly attach the name to the data.

An NDN packet performs best effort data recovery," just like IP packet delivery . Data or interest packets may be lost during processing. Therefore, it is the end user's obligation to re-transmit the interest back to the network if the consumer does not get the necessary data following the predicted RTT. However, NDN packets transmit the data names rather than the source and destination, in contrast to IP's location-centric data delivery strategy.

Even though they are fairly minor design variations, they result in two significant process profound changes. First, NDN customers lack the names and addresses needed to deliver data packets. The NDN routers take its place by recording every interface that comes in and using data from pending interest to return the relevant information to the customer.

Second, although the interest packet's name directs the forwarding process, similar to how the destination address directs the forwarding of an IP packet, the interest may locate a copy of the requested data in a nearby router and return the data to the customer, whereas an IP packet travels and reaches the destination.

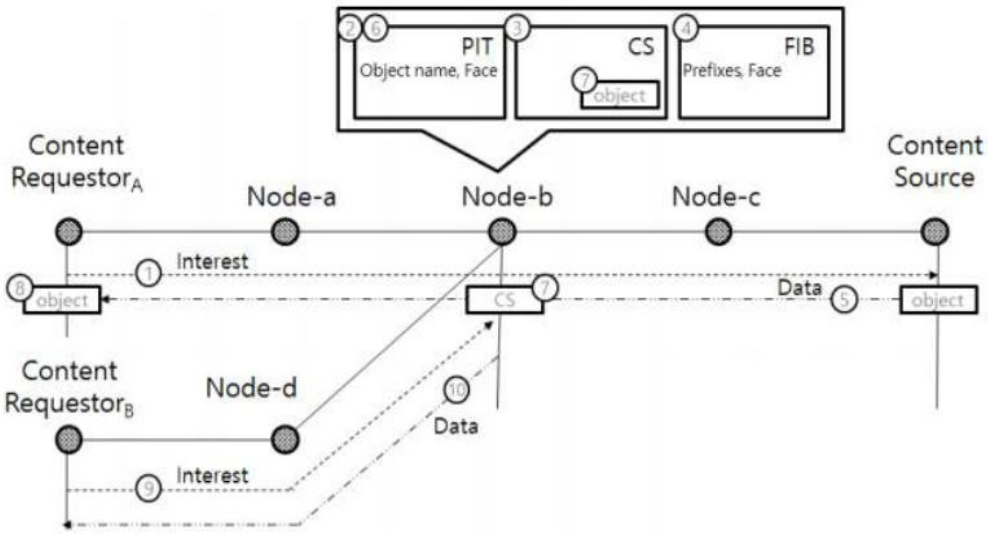

Figure 4 Data Networking Architecture (Ndn) Interest/Data Procedure [23]

As seen in Fig. 5, NDN came from the Information Centric Networking (ICN) research field, which served as the model for several subsequent Internet designs. Researchers have recently examined the main characteristics and problems of NDN as the Internet architecture of the future (FIA). The design ideas of NDN have been explored in relation to various FIAs, including AKARI, JGN2Plus, FIND, NEBULA, XIA, GENI, 4WARD , FIRE, and others.

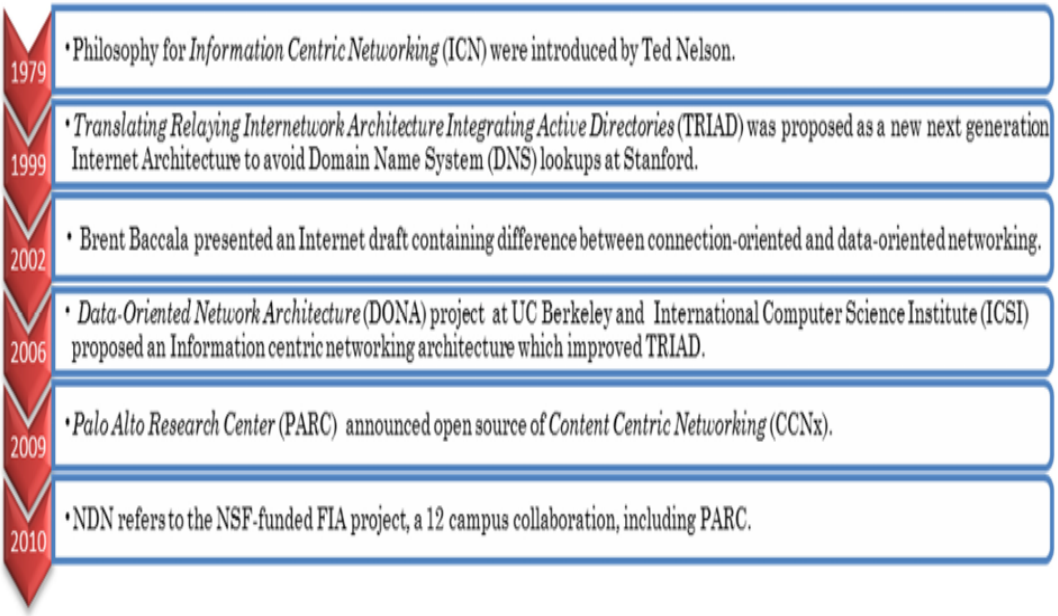

Figure 5 NDN Timeline [17]

6.2 NDN Architecture :

6.2.1 The main entities in NDN :

- **Producer**:

The producer identifies the entity that creates or generates the data. It gives this data a distinct name and posts it on the NDN network. The producer can reply to a consumer's request for a particular dataset by providing the data linked to that name.

- **Consumer**:

In the NDN, the entity that makes the initial request for data is the consumer. Instead of referring to IP addresses and specifying the source of the data, the consumer makes an "Interest" query using the name of the data they are looking for. NDN routers utilize this name to find and send the relevant data to the customer when the request is broadcast over the network.

- **Router**:

NDN routers base their routing decisions on name information. A router utilizes the request name to identify the location of the data when it gets a consumer request. Likewise, when a router gets information from a producer, it determines how to forward that information to customers who have submitted matching requests based on the linked name. In the NDN network, routers are crucial for controlling data caching and enabling search and routing.

6.3 Packets types in NDN

Each NDN node consists of three main components:

- **Interest packets**: The requesting nodes send these packets in order to request certain data. A packet of interest including the name of the desired content is sent by a node when it wants to obtain a specific piece of content. In an attempt to find the relevant data, this packet is sent across the network.
- **Data packet:** In response to a packet of interest that matches content that it possesses, a data-holding node sends a data packet with the requested content. These data packets can be stored in the network for subsequent use and are signed by the content creator to guarantee their legitimacy.

NDN relies on these two kinds of packets as its basic communication mechanism. The requesting nodes send packets of interest to request specific data, and the data-holding nodes reply by sending matching data packets.

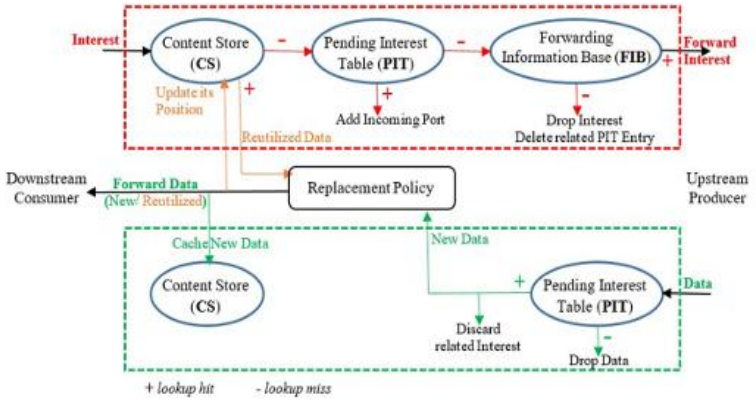

Figure 6 - The NDN router's processing for Interest and data packets [24]

6.4 Key Data Structures in NDN

- **Pending interest table (PIT):** All of the concerns that a router has sent but has not yet addressed are included in the Pending Interest Table (PIT). The data name transmitted over

the Internet, together with its incoming and outgoing interface(s), is recorded in each PIT entry [28]. When a new interest packet arrives, an item is added to the Pending Interest Table, which keeps track of unmet interests. It is deleted when the matching data packet satisfies it [29].

- **Forwarding information base (FIB):** A transport table called the Forwarding Information Base associates interfaces with ingredient names. Similar to IP, forwarding information bases are utilized to deliver interest packets based on the longest prefix match [30]. A name-prefix based transport protocol populates the Forwarding Information base, which may have a distinct output mediator for every prefix.
- **Content store (CS):** The Content Store is a cache that saves preprocessed data packets when they are reordered. The Content Store serves as a temporary store for data packets received by the router and can be delayed to satisfy future needs because an NDN packet has significance regardless of its source or redirection. Although the replacement approach has been employed recently, it is decided and may differ by the router.

6.5 Communication in NDN :

In NDN, communication is driven by the receiving end, often known as the data consumer. A consumer sends out an Interest packet with a name that specifies the requested material in order to receive it. A customer may ask for /parc/videos/WidgetA.mpg, for instance. After remembering the interface from which the request originates, a router uses its Forwarding Information Base (FIB), which is filled with information from a name-based routing protocol, to forward the Interest packet. When the Interest reaches a node with the requested data, it sends back a Data packet containing the name and content of the data, as well as a signature by the producer's key .The path that was established by the Interest packet is traced back to the customer by this Data packet. It should be noted that neither the Interest nor the Data packets contain any host or interface addresses (such as IP addresses); instead, the Interest packets' names are used to route them to the data producers, and the Interests' state information at each router hop determines how the Data packets are returned

For a while, ND routers keep both data and interests. Only the first Interest is transmitted upstream in the direction of the data source when many Interests for the same data are received from downstream. The Interest is then saved by the router in the Pending Interest Table (PIT), where each entry includes the Interest's name and a list of interfaces from which the corresponding Interests were obtained. The router transmits the data to every interface specified in the PIT entry after locating the corresponding PIT entry when the data packet arrives. After deleting the relevant PIT item, the router stores the data in the Content Store,

which is essentially its buffer memory that is governed by a cache replacement policy. Though in the opposite direction, data follows the same course as the interest that requested it. Hop-by-hop flow balance is achieved when one Data fulfills one Interest each hop.

The router can store an NDN data packet to meet possible future requests since it has significance regardless of its origin or destination. This allows NDN to automatically support a number of features without the need for additional infrastructure, such as multicast (many users requesting the same data simultaneously), mobility (users requesting data from different locations), delay-tolerant networking (users with intermittent connectivity), and content distribution (many users requesting the same data at different times).

Let's take the example of a customer watching a streaming movie while driving. After making a data request, the customer could switch to a different local network. The data is cached along the way, but it will eventually arrive at the previous place and be discarded. The disruption will be minor since the consumer will probably retrieve the data from a nearby cache when it retransmits the interest. Data cached near users enhances packet delivery efficiency and lessens reliance on a specific data source that might malfunction as a result of errors or intrusions.

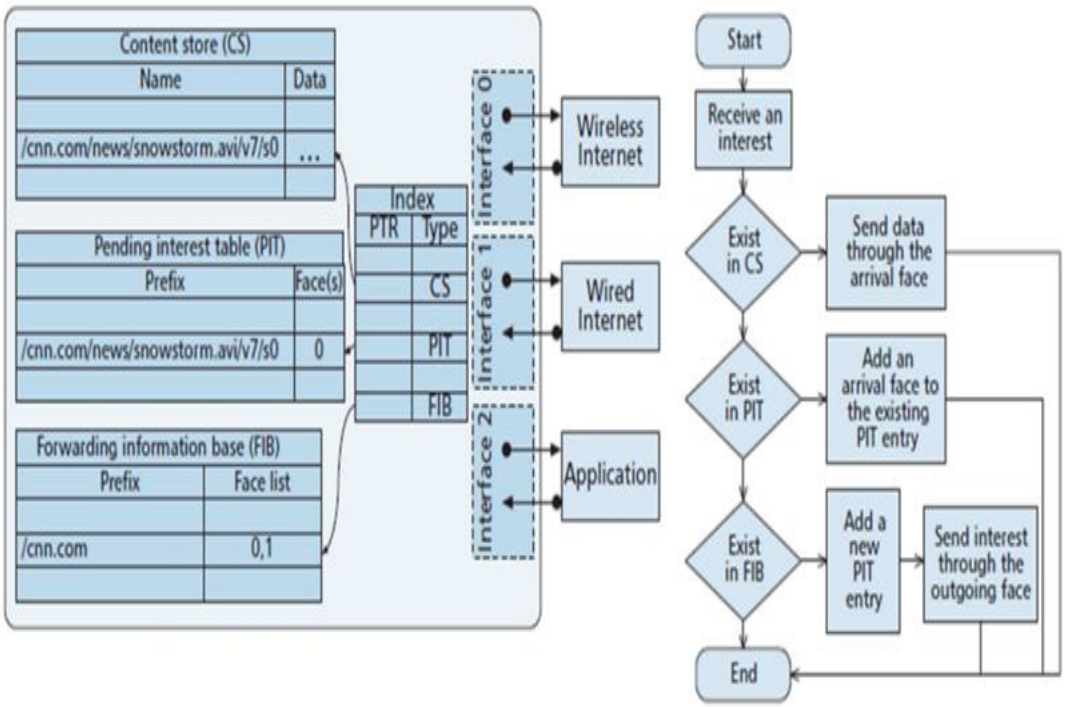

Figure 7 Communication process in an NDN node [27]

6.6 NDN vs IP

6.6.1 Benchmarking between PI and NDN

Functionality	IP architecture	NDN architecture
Communication	Based on host addresses (where to send)	Based on the names of the bare contents (what to send)
Routing	By destination address	Par nom de contenu
Security	By secure channel (TLS, IP Sec, etc.)	By signed content (digital signature, etc.)
Caching	By proxy or CDN	By router
Mobility	By address redirection (Mobile IP, etc.)	By reexpression of interest
Multicast	By specific protocol (IGMP, PIM, etc.)	By selective distribution
Latency	Depends on distance and number of jumps	Depends on the popularity and location of the data
Debit	Depends on bandwidth and congestion control	Depends on data availability and congestion control
Packet Loss	Can be caused by transmission errors or congestion	May be caused by unrelated interests or interests

6.6.2 The advantages of NDN over IP networks

- By offering quicker streaming, shorter buffering periods, and a better user experience, NDN can increase the effectiveness of content delivery.
- Potential advantages of using NDN in healthcare settings include enhancing the speed at which medical personnel can get data, maintaining confidentiality, and safely managing vast volumes of patient data.
- By concentrating on content rather than location, NDN can assist alleviate network congestion. Named Data Networking (NDN) lessens the strain on certain network paths by more effectively spreading popular material across several nodes.
- By facilitating quicker access to frequently requested material, lowering latency, and enhancing network performance generally, caching is essential to enhancing the NDN user experience.

- The data security mechanism of NDN eliminates the need for software and physical isolation and secures communication channels, guaranteeing the security of all data generated during its lifespan.
- NDN eliminates the requirement to set up networks with IP addresses by sending data and packets of interest using application layer names. When there are several linked devices, this simplicity is very helpful.

6.7 The main features in NDN

- **Naming**: Named data objects (NDOs) are used by NDNs to represent a variety of material, including photos, videos, web pages, and more. Often hierarchical, naming enables distinct content identification.
- **Name-based routing**: Name-based routing, which uses content names rather than host addresses to route content requests, is a feature of NDNs. High responsiveness in the case of abrupt network changes is made possible by this.
- **Caching:** One of the main components of the NDN strategy is data caching within network nodes. This lowers server loads and response times, which enhances the performance of content delivery.
- **Content Security:** In NDN networks, priority is given to ensuring the authenticity and integrity of content. This is done in part by digitally signing the matches between names and content to ensure data security.
- **Decentralization of information:** NDNs make better use of network resources and enable more effective content distribution by concentrating on the content itself rather than where it is located on the internet.
- **Improved network efficiency:** As user bandwidth needs rise, NDN design seeks to enhance network scalability, efficiency, and content delivery. These features outline the NDN architecture's fundamentals and show how this paradigm aims to get beyond the drawbacks of the existing Internet model. To do this, it prioritizes caching, security, and content.

6.8 Names:

NDN names are opaque to the network, meaning that routers are aware of the boundaries between components in a name but not its meaning. As a result, naming schemes can change independently of the network and each application can select the one that best suits its requirements.

A movie created by PARC may have the name/parc/videos/WidgetA.mpg, where the '/' denotes a boundary between name components (it is not part of the name). This is an example

of how NDN design assumes hierarchically organized names. Applications may effectively depict the relationships between data points by using this hierarchical structure. Segment 3 of version 1 of the video, for instance, may be called /parc/videos/WidgetA.mpg/1/3.Routing may also be scaled thanks to the hierarchy.

Although routing on flat names may theoretically be feasible, aggregation is made possible by the hierarchical structure of IP addresses, which is crucial for scaling to the current routing system. Conventions agreed upon by data producers and consumers, such as name conventions signaling versioning and segmentation, can provide the common structures required to enable programs to work over NDN names. Name conventions are network-invisible and application-specific.

Global uniqueness is not a requirement for names, but it is necessary to retrieve data internationally. Names meant for local communication could rely mostly on local context and only need local broadcasting or local routing to locate relevant information.

Customers must be able to deterministically create the name for a requested piece of data without having seen the name or data before in order to obtain dynamically produced data. Either (1) consumers can get data based on incomplete names, or (2) deterministic algorithms enable producers and consumers to arrive at the same name based on data that is available to both. A data packet with the name /parc/videos/WidgetA.mpg/1/1 might be returned to the user, for instance, if they request /parc/videos/WidgetA.mpg. Using the information provided by the initial data packet and the naming scheme decided upon by the producer and consumer apps, the consumer may then request and define further segments.

The naming system is the most crucial component of the NDN architecture and is still being researched; specifically, it is currently unclear how to establish and assign top level names. Not all naming issues must be resolved right away, though; because names are opaque to the network and rely on applications, the design and development of the NDN architecture can—and should—occur concurrently with our investigation into name structure, name discovery, and namespace navigation within the framework of application development. Here are some specifics on the NDN naming:

a) Hierarchy: NDN names can be layered within one another to create a tree structure since they are hierarchical. A directory structure can be used to arrange names, with higher directory names holding files and subdirectories.

b) Tree structure: NDN names are derived from a tree structure, in which each node denotes a name element. Information about the material, like its kind, publisher, publishing date and time, and location, might be included in the name.

c) Names in NDN are unique, meaning that a resource may be uniquely identified by its name. By employing version IDs, data editors may guarantee that each version of the data has a distinct name.

d) Users can obtain certain portions or versions of a piece of data instead of the complete thing by utilizing segment and version designations in the data names. This can increase content delivery via NDN networks' dependability and efficiency.

6.9 Routing and forwarding

In the present "IP" structure, four issues have been resolved: mobility, "NAT" traversal, accessible report management, and space enervation. The aforementioned four issues with the "IP" structure are eliminated by "NDN's routes and forwards." Routing can be done in the same manner as "IP" routing nowadays [28]. The title prefaces that a router announces instead of "IP" precede the data that the router is ready to serve. This declaration is transmitted via a routing protocol. Every router constructs its "FIB" based on the routing proclamations it has received. It is possible to adapt conservative routing systems like "OSPF" and "BGP" to route based on name prefaces [29]. Names are handled by routers as an impenetrable module order. They only match the "Content Name" from a pack "component-wise" in terms of preface length against the "FIB."

In the "FIB," for example, /work/update/info.pdf may compete with both /work/update and /work. The fact is that "NDN" inherits the ability to implement multipath routing. To prevent circles, "IP" routing admits just one optimal path. Because the name and an accidental nonce can effectively identify and eliminate duplicates, it means that "NDN" interest cannot loop in an advantageous way. Since they follow the opposite path of benefits, data do not circle. By doing this, the "NDN router" may use the many boundaries to raise an issue without worrying about the loops. The first piece of information that is returned will allay the worry and be gathered locally. Duplicates received later will not be accepted. The routing security mechanism of the "NDN" has been significantly enhanced. first, which has routing posts, protects them from being fooled or tampered with. Second, multiple pathways routing prevents preface overthrow. In the meanwhile, routers can detect irregularities caused by preface takeover and try to retrieve the data in other ways. Third, the "NDN" communications may actually only be sent regarding records. Directing malicious packages to a certain mark is challenging since it simply cannot be communicated to many different kinds.

6.10 Caching

In NDNs, caching is the practice of locally storing the most requested named data to provide faster access to it later. The first thing a network router does when it receives a request for specified data is determine if it already has that data in its local cache. If so, the router can reply to the request straight from its cache, which lowers network load and delay.The Forwarding Information Base (FIB) and Pending Interest Table (PIT), which are crucial elements of query routing and processing in NDNs, are intimately associated with caching. Usually, factors including data size, request frequency, and cache management guidelines are taken into consideration while deciding whether to cache named data.

Entries in the PIT table, which lists pending requests of interest for that data, are frequently linked to cached data. The node can additionally update the PIT table in accordance with the cached named data, resolving any outstanding queries of interest. The FIB table, which keeps track of routing information to decide how data packets should be sent to their destination, also has an impact on caching.

The FIB table's entries show the further steps to take in order to get to the nodes that can supply the needed data. As a result, the node can update the FIB table to forward future requests either to its own cache or to other network nodes that are storing the required data when a particular piece of data is cached.

Additionally, by decreasing the chance of data loss from outages or connection problems, caching contributes to increased network resilience. As a result, caching in NDN networks is essential for enhancing network dependability, efficiency, and performance.

The following four metrics are typically used to measure cache performance: hit ratio, content retrieval delay (the total amount of time that passes between the time a content request is generated and the time that the consumer receives it), and average number of hops traversed (the number of hops required to locate and retrieve a requested piece of content is also measured to gauze how well the content is distributed across the network). Dissemination speed is another helpful cache performance parameter that quantifies the amount of time needed to distribute material all the way to the network edge**.**

The cache decision policy specifies whether or not to cache the data packet at the intermediate nodes. Two key concerns for a successful caching algorithm are: where should content be cached? What material has to be replaced first? Therefore, we divided caching systems into two general categories: cache replacement (deciding whether to store content on the router) and cache placement (deciding whether to place content on the network).

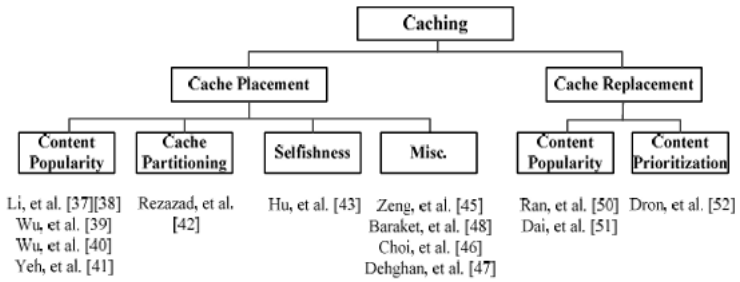

Figure 8 Classification of different caching strategies

6.11 Cache placement

The NDN architecture includes a cache placement mechanism called Leave Copy Everywhere. In LCE, a data packet is cached by each router that sits between the producer (or provider) and the consumer. This technique's high cache redundancy, which is achieved by having the identical item cached across numerous nodes, lowers the system's total cached content variety. Two popular cache placement strategies to reduce cache redundancy are Leaving Copies with Probability (LCProb) and Leaving Copies with Uniform Probability (LCUniP). LCProb employs caching probability 1/ (number of hops) for caching content on the router, whereas LCUniP employs uniform probability [30]. The cache variety of the whole network will be improved and content download latency will be reduced by storing popular content on the network edge, which will maximize the utility of cached information across the system. Reducing cache redundancy and increasing cache diversity also requires an efficient router coordination strategy.

6.12 Cache replacement

Least Recently Used (LRU) is a popular cache override strategy that works well and raises the likelihood of a cache hit by temporarily storing the most recent data. Least frequently utilized (LFU), which removes the least utilized stuff first, is another crucial cache replacement policy. The content's arrival on the router and its replacement can be used to determine when to make a caching choice. Content can be relocated one level upstream in the cache hierarchy for caching, but it shouldn't be deleted from the cache for improved network performance. Less popular and low-priority material is replaced first in cache replacement, which is divided into two categories: content popularity and content prioritization.

7. Conclusion

The internet's original architecture, which was not intended to handle the rapid growth in usage and information, is the cause of its current issues and challenges. The researchers suggest a new ICN Internet architecture as a result.

One particular use of ICN that is thought to be a potential strategy for communication networks in the future is NDN. The goal of this next-generation network is to enhance internet data transmission. NDN places more emphasis on content-based communication than IP addresses, in contrast to the present Internet, which is based on IP addresses and is primarily intended to transport data between two hosts.

Chapter-II-

AI-based cache replacement polices

1. Introduction

Efficient cache management is critical for improving system performance, especially in contexts with limited memory resources and variable data access patterns. Cache replacement policies play an important role in choosing which data items should be kept in the cache and which should be evicted when new data arrives and the cache is full.

Traditionally, a number of cache replacement methods have been created, each based on a different criterion, such as recent access, frequency of usage, object size, or a mix of these. Traditional policies such as Least Recently Used (LRU), Least Frequently Used (LFU), First-In-First-Out (FIFO), and size-based techniques are all intended to increase cache hit rates while minimizing latency within the limits of limited capacity. While these techniques have shown to be useful in a variety of contexts, their static nature limits their capacity to adapt to changing access patterns and complicated workloads.

In recent years, intelligent cache replacement strategies have emerged, ushering in a new cache management paradigm. These current solutions, which use improvements in machine learning and data-driven optimization, allow caches to dynamically adapt to changing data access habits and system needs. Intelligent policies can use predictive analytics, reinforcement learning, and hybrid models that learn from past access patterns to make better replacement decisions. As a result, intelligent cache replacement algorithms have shown significant improvements in cache efficiency, hit ratios, and overall system responsiveness, particularly in complex or large-scale computing systems. This chapter examines both standard and intelligent cache replacement policies, emphasizing their concepts, strengths, and the advances brought forth by intelligent techniques.

2. Traditional cache replacement policies

Caching performance in NDN networks depends on two factors: content placement method and replacement policy. In this context, we highlight this part for Enhancing Caching Performance in Named Data Networking that has been proposed or enhanced to increase overall network performance. There are three sorts of data replacement policies: those based on popularity, those based on recency, and those that take into account both [47, 48].

3. The Least frequently used (LFU):

It was suggested that NDN networks adopt the LFU policy. The premise behind this policy is that content that is frequently requested may shortly be requested again. In order to make room for new content, the method entails removing the least popular items from the cache.

However, even if not specifically requested, very popular content may persist in the CS, creating a problem known as "aging phenomenon" or "cache pollution." To address this problem and improve the performance of the entire network, a number of LFU variations have been proposed, such as LFU-aging, LFU with dynamic aging, and Window-LFU [49,52].

LFU-Aging [37] is an extension of LFU. This policy minimizes cache pollution, which is the cache of popular content that is no longer accessed. LFU-Aging uses a threshold; if the value of all counters is above this threshold, they will all be halved. In [51], authors proposed (LFUda) LFU with dynamic aging , as an enhancement to LFU-Aging to mitigate the risk of cache pollution.

4. The Least Recently Used (LRU):

LRU (Least Recently Used) is a cache replacement algorithm that works by replacing data that hasn't been accessed for a long time. The main idea behind LRU is to assume that data that has been accessed recently is more likely to be accessed again in the near future. The LRU algorithm maintains an ordered list of data based on its last access, so that the least recently used data is replaced first when new data needs to be cached. LRU is widely used because of its simplicity and ability to exploit the temporal locality of data accesses [53].

5. The Least Recently/Frequently Used (LRFU):

The LRFU strategy for NDN networks combines LRU and LFU strategies, as presented in [54]. At full capacity, LRFU chooses the data with the lowest CRF (combined recency frequency) value for eviction. The experimental results showed that the LRFU replacement policy had a 3.36% higher hit rate than the LRU and 5.78% higher hit rate than the priority-FIFO replacement strategy [52].

6. The Window LFU (WLFU):

It was originally implemented for web caching in [55]. The WLFU Cache Override Policy, is an enhancement on the LFU algorithm that takes into account the timing and popularity of cached objects. Here are some crucial points about the WLFU replacement policy.

WLFU keeps a sliding window of the most recent requests and utilizes the LFU algorithm to replace objects in the cache based on their popularity inside that window. When numerous items have similar popularity, the LRU algorithm is used to determine which object should be replaced. By adjusting the window size suitably, this override policy can effectively balance

the benefits of temporality and popularity, improving cache success rates. Thus, WLFU combines the temporality and popularity of object accesses, potentially leading to better performance than existing cache replacement techniques.

7.The Two-Queue (2Q):

The two-queue (2Q) replacement policy was introduced in [56]. This caching strategy seeks to improve the LRU policy by adhering to the principle that when a content really requires caching, it should regularly receive periodic requests, particularly after a large number of accesses in a short period of time. As a result, the 2Q policy oversees a FIFO queue and two LRU lists. In actual applications, the 2Q method

8.The adaptive replacement cache (ARC) policy:

The adaptive replacement cache (ARC) policy [57] improves LRU performance by managing two LRU lists of variable sizes: T1 for data retrieved once and T2 for data retrieved on a minimum of two occasions. Additionally, ARC only saves the names of recently evicted data from both lists, not its contents.. It makes use of two extra LRU lists: B1 for managing content newly removed from the T1 cache and B2 for handling content recently removed from the T2 cache. The results demonstrate that ARC has a 4% greater hit rate than the LRU replacement policy [52].

9. Intelligent cache replacement policies

9.1.General introduction to Machine Learning and Deep Learning

9.1.1. Machine learning :

Machine learning (ML) is a subfield of artificial intelligence (AI) that seeks to enable machines and machines to learn like humans, execute tasks independently, and improve their performance and accuracy as they gain experience and exposed new data. Four types of machine learning tasks can be described:

- ***Supervised learning:*** Trains models on labeled data to predict or classify new, unseen data.
- ***Unsupervised learning:*** Finds patterns or groups in unlabeled data, like clustering or dimensionality reduction.
- ***Semi supervised learning:*** uses both labeled and unlabeled data, making it helpful when labeling data is costly or time-consuming

- ***Reinforcement learning:*** Learns through trial and error to maximize rewards, ideal for decision-making tasks.

These four task types have significantly improved when working with high-dimensional data, including time series, pictures, and videos, primarily because to the recent advancements in deep learning.

Deep learning is gaining popularity due to three complementary factors: increased computational power from GPUs, methodological breakthroughs [46], and a growing ecosystem of software and datasets.

9.1.2. Deep Learning:

Deep learning originated as a model of neural processing in biological brains. Deep learning may not align with current neurobiology knowledge [44], but there are some parallels, such as convolutional layers inspired by the animal visual cortex [45]

An artificial neural network, often known as a neural network, is, in its most basic form, a function $f: X \rightarrow Y$ parameterized with θ that accepts x ∈ X as input and returns y∈ Y as output (X and Y depend on the application):

$$y = f(x; \theta)$$

Inside a deep neural network, information passes from the input layer, which receives raw data such as photos, text, or audio, through a number of hidden layers composed of linked artificial neurons. Each neuron processes its inputs by adding learnt weights and biases, passes the result via an activation function to induce non-linearity, then transmits the output to the following layer.

As input progresses further into the network, each hidden layer extracts more abstract features; for example, in picture recognition, early layers may identify edges, while deeper layers recognize forms and objects. During training, the network creates predictions, compares them to the real responses with a loss function, and then adjusts its internal weights via back propagation to reduce mistakes. This repeated approach allows the network to improve its knowledge and increase accuracy over time. Deep learning's layered design and capacity to learn directly from raw data have led to advancements in domains including computer vision, speech recognition, and natural language processing, making it a cornerstone of current artificial intelligence.

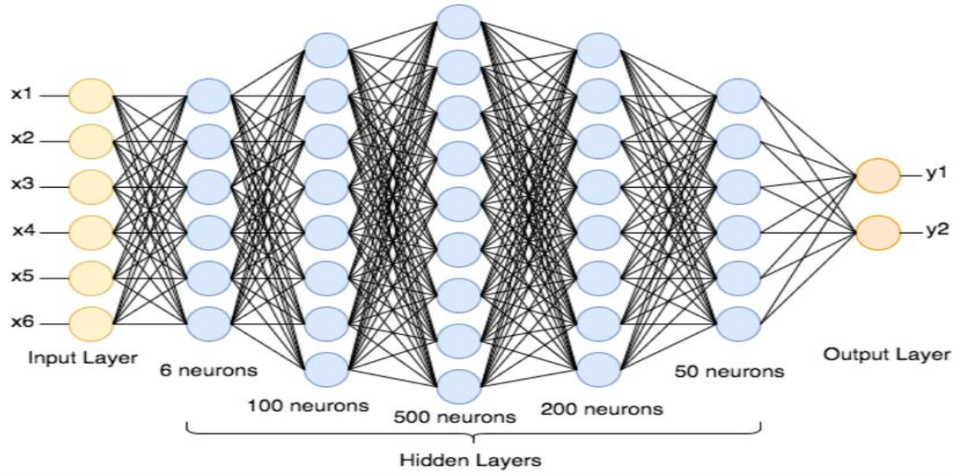

Figure 9 deep neural network [62]

9.1.3. Reinforcement learning

9.1.3.1. Introduction

The field of machine learning known as reinforcement learning focuses on teaching agents to behave in unfamiliar situations by using reward signals. Through repeated action selection, an agent must learn to optimize cumulative rewards from the environment. The issue turns into a recurring decision-making difficulty. After assessing the environment's present condition, the agent acts. Following the action, the environment gives the agent a reward and a new state. Until a terminating event occurs, this procedure is repeated as in (**figure**). The best course of action, such as maximizing the reward from the environment at each step, must be learned by the agent. Markov Decision Processes (MDPs) are commonly utilized to give a mathematical framework for describing the situation in order to formalize the reinforcement learning environment [35].

MDPs are a way to formalize decision-making processes in which an agent learns to behave in a way that maximizes reward through repeated interactions with the environment. The agent must repeatedly choose what to do in an MDP.

An MDP is defined as the set of states the environment can be in S, the set of actions the agent can do A, and the transition function P(s'| s,a): S x S x A→ $\mathbb{R}$, where s,s' ϵ S and a ϵ A is the reward function R(s,a): S x A → $\mathbb{R}$, where s ϵ S and a ϵ A, and a discount factor with $0 < \gamma < 1$. The transition function represents the probability of transitioning from one state to another if the agent takes some action. The reward function returns a scalar reward value for doing an activity in the current environment.

One significant property of MDPs is their adherence to the Markov Property. According to the Markov Property, the future state of an environment purely depends on the current state. When the present state is known, previous states have no effect on future states or rewards. This assumption is usually correct, but it is occasionally violated.

To further understand MDPs, consider the following scenario. During time steps

t = 0,1,2,..., the agent observes a state st S from the environment, takes an action at A, obtains a reward rt R, and moves to the next state st+1 depending on the transition probability. This can be written as a sequence of occurrences.

$$\boldsymbol{s0, a0, r0;\ s1, a1, r1}$$

The agent's purpose is to maximize the cumulative reward of this set of actions or∑t rt. To accomplish this, the agent needs to learn a policy function π(s): S→A This informs the agent on the appropriate action to take based on the environment's state. The function is written sometimes π(a| s): S x A→ℝ which defines a function that tells the agent the probability it should take action given its current state.

$$\boldsymbol{\pi(a|\, s) = Pr\,(action = a|\, state = s)}$$

However, solving for this policy function is difficult for a number of reasons. Often, certain aspects of the MDP remain unknown. The transition function P(s'| s,a) is often stochastic and not completely defined. After performing an action in state s, it is not always predictable which state the environment will end up in. This normally follows a random process with unknown probabilities. The reward function, R(s,a), might be either partially known or stochastic. The agent is unaware of the reward it will receive from the environment upon taking an action. The unknown aspects of the MDP make determining the appropriate policy problematic. Because these aspects of the MDP are unknown, determining the best policy is impossible. Instead, approaches must be utilized to try to learn the optimal policy through interaction with the environment.

Solving a stochastic MDP can be divided into two types. The first group involves model-based approaches. To solve the MDP, these algorithms attempt to directly learn the transition function P(s'| s,a) and the reward function R(s,a). Model-free approaches aim to identify a policy in an unknown environment rather than learning a model of it.

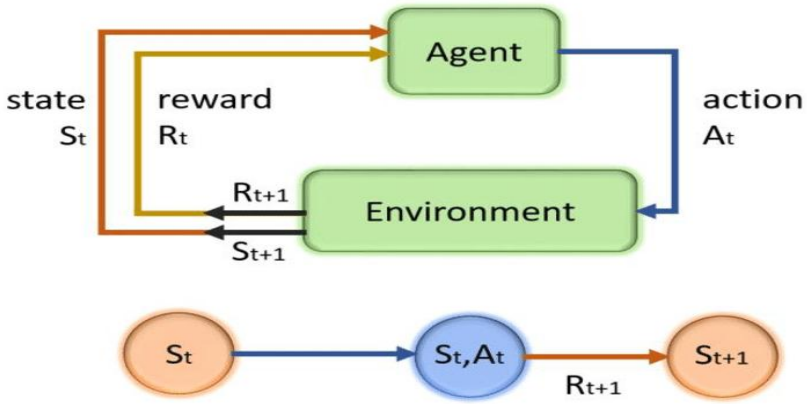

Figure10 standard architecture of RL

9.1.3.2 Reinforcement learning approaches:

9.1.3.2.1 Model-based Learning :

Model-based learning is a method of RL in which an agent employs previously acquired knowledge to execute tasks, where the agent will have a complete description of the environment (transition probability "dynamics of the environment). This method is distinguished primarily by the employment of a model that describes the dynamics of the environment, i.e. how the agent's actions influence the state of the environment and the rewards it can get. This sort of learning is classified into two major categories:

9.1.3.2.1.1 Learn the Model and **Given Model** [58].

a) Learn the model:

The agent in many real-world scenarios is initially unaware of how the environment functions, meaning it is unaware of the rules governing state transitions or the rewards it will receive for taking actions. In this case, the agent must learn the model by interacting with the environment, observing what happens when it takes actions, and gradually building an internal representation (a model) of the environment's dynamics.

b) Given model:

Occasionally, the environment's dynamics are already known or provided to the agent. This indicates that the agent has access to a given model-a complete description of how actions lead to new states and rewards. With this information, the agent can directly plan the best

actions without needing to learn the model from scratch. This is typical in simulated environments, classical control problems, or games where the rules are fully specified.

c) Model-free:

A model-free approach is one that does not estimate the transition probability distribution (or reward function) associated with the Markov decision process (MDP),[31] which represents the issue to be solved in RL. The transition probability distribution (or transition model) and the reward function are commonly referred to as the environment's "model" (or MDP), thus being named "model-free". A model-free RL method can be compared to a "explicit" trial-and-error approach. In other way It's when the agent does not know the model dynamics of its environment (transition probability).

d) Policy based

The agent directly learns the policy that determines the actions to be taken in each state without going through an explicit estimation of the values of the states or actions. They primarily use Monte Carlo techniques to estimate gradients in politics from trajectories of complete episodes.

e) **Value based**

In this approach, the agent learns to value actions in terms of value, usually represented by an action-value function (s,) and it is divided into two sub branches:

- **On policy:** are algorithms that use the agent's current policy to make decisions and to update its knowledge of the environment by learning from the actions selected by it. This means that the agent follows their current policy to explore the environment and to learn from these experiences. A common example of an on-policy algorithm is Sarsa using Time Difference Learning (TD).
- **Off policy**: are algorithms that conversely use a different policy than the agent's, allowing for more informed decisions to be made and more diverse behavioral data to be collected in order to update their knowledge of the environment. This means that the agent follows a different policy than the one it uses to explore the environment. A common example of an off-policy algorithm is Q-learning.

This section focuses on model-free algorithms, which are more relevant to the problem at hand. In model-free approaches, policies are learned to maximize cumulative reward. The

value function of a policy dictates how much reward will be obtained for all subsequent time steps. This can be officially spelled as:

$$V^{\pi}(s_t) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s\right]$$

It represents the expected reward from following policy starting in state s. Reinforcement learning seeks to identify a policy that maximizes the value function for all states.

9.1.4 Markov decision process:

9.1.4.1 Definition

Also known as a stochastic dynamic program or stochastic control issue is a model for making sequential decisions when the consequences are unknown [31].

MDPs originated in operations research in the 1950s [32][33] and have subsequently garnered attention in a range of sectors, including ecology, economics, healthcare, telecommunications, and reinforcement learning. [34] Reinforcement learning uses the MDP framework to simulate the interaction between a learning agent and its environment. In this paradigm, interactions are defined by states, actions, and rewards. The MDP framework is intended to give a simpler depiction of important aspects of AI difficulties. These aspects include comprehending cause and effect, managing uncertainty and no determinism, and pursuing defined goals.

The term "Markov decision process" is derived from the Russian mathematician Andrey Markov's concept of Markov chains. The "Markov" in "Markov decision process" refers to the underlying structure of state transitions that still follow the Markov property. The process is called a "decision process" because it entails making decisions that influence these state transitions.

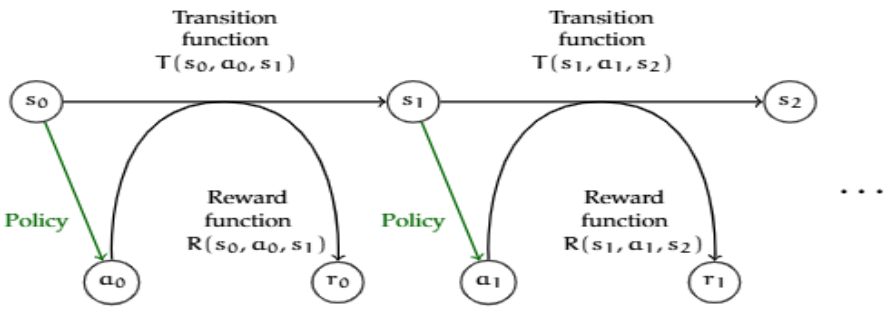

Figure 11 illustration of an MDP.

9.1.4.2 The Markov property:

The Markov property is a key notion in probability theory and stochastic processes that expresses the idea of "memory lessness ". It clarifies: If the agent is present in the current state s1, performs an action a1 and move to the state s2, then the state transition from s1 to s2 only depends on the current state and future action and states do not depend on past actions, rewards, or states."

Orin another words, according to the Markov Property, the present state transition is independent of any previous action or state. Thus, MDP is an RL issue with the Markov property. In a chess game, for example, the players only need to remember the current state and not previous actions or states.

9.1.5 Return:

Since the goal is to maximize the reward over all time steps using an optimal policy – i.e., the expected return – we also need a way to calculate this value. Naively, we could define this reward using the sum of the sequence of rewards:

$$\boldsymbol{R(\tau) = r0 + r1 + \cdots + rT}$$

However, for some tasks where the agent continuously undertakes actions without an end-state, this approach will fail because the sum of all rewards will tend to infinity when $R(\tau) = \infty$. This would prevent the agent from distinguishing actions that produce larger rewards more quickly.

That's why we're introducing a factor of γ discount ($0 \leq \gamma \leq 1$) that is used to weight the importance of future rewards versus immediate rewards.

The total expected return is then calculated as follows:

$$\boldsymbol{R(\tau) = t = 0 \sum \infty \gamma t r t}$$

- When γ is close to 0, more importance to immediate rewards than future rewards.
- When γ close to 1, more importance to future rewards than immediate rewards.
- $\gamma = 0$ → agent not learn, $\gamma = 1$ → agent won't stop learning.

If the agent knew the precise sequence of rewards they would get, no additional calculations would be necessary. However, the values of the rewards at each stage depend on both the state of the environment and the action chosen by the agent. For this reason, we define an action value function that helps us approximate the values of specific state-action pairs.

9.1.6 Value function:

Is also called state value function it shows the importance of existing in that state. It gives information about how well the situation and action are and how much reward an agent can expect. A reward represents the immediate signal for each good and bad action, whereas a value function specifies the good state and action for the future. The value function depends on the reward as, without reward, there could be no value. The purpose of estimating values is to achieve more rewards.

- **The state value function $V^{\pi(s)}$:** It estimates the expected return based on state s while always following policy π.

$$\boldsymbol{v\pi(s) = E\pi[Rt|st = s] = E\pi[t = 0\sum\infty\gamma tRt + 1 \mid s0 = s]}$$

- **Actin value function $Q^{\pi(s,a)}$:** IT estimates the expected return starting from state s, by taking action a, and then always following policy π.

$$\boldsymbol{Q\pi(s,a) = E\pi[Rt \mid s0 = s, a0 = a]}$$

9.1.7 Optimal value function:

The optimal value function V*(s) yields the maximum value compared to all the other value functions.

$$\boldsymbol{V*(s) = max\pi\, v\pi(s)}$$

9.1.8 Cache Replacement Policy as a Markov Decision process:

The cache replacement policy can be modeled as a Markov Decision Process (MDP). This research models the MDP agent as an algorithm that determines which cache elements should be evicted when full. The problem of cache replacement lends itself well to treatment as a Markov Decision Process. Caching involves retrieving a set of data items from their original source. As various pieces of data are requested, some of them are saved in the cache for later use. However, once the cache is filled, certain items must be removed. When the cache is full and a new data request is received, it must continually decide which item to discard. This type of recurring decision process is precisely what MDPs are intended to mathematically represent. The agent in this Markov decision process acts as an internal cache agent, deciding which elements to preserve and which to remove.

The Markov Assumption must also be true for an issue to be a good fit for modeling as a Markov Decision Process. The cache's current state is determined only by its contents. The cache's current state is unaffected by its prior contents. Furthermore, the cache's present state is unaffected by the previous data items that it removed. Thus, the Markov property is true

for cache replacement policy. This demonstrates once more how obviously well-suited this situation is to be an MDP.

State: The possible states that the cache can be in are represented by the state space S of the MDP.

Action: At each cache miss, the agent must pick an action—typically, which content item to evict to make way for the new one.

Reward: A reward is given to the agent (for instance, a cache hit results in a positive reward, but a miss results in a zero or negative reward). The goal is to maximize the predicted cumulative reward over time, which is frequently achieved by lowering latency or increasing the cache hit ratio.

This Markov decision process has a stochastic and uncertain transition function, P(s'| s,a). The shift from one cache state to the next is determined by next pieces of data requested in the data access pattern. When one item of data is removed from the cache, the next piece of data takes its place, resulting in a succession of fresh data requests. The state is determined by the next set of data requested and used. This cannot be predicted ahead of time, hence this MDP is not fully known.

The discount factor in this MDP is set to be close to one. The series of data accesses is typically long and each reward is not unduly dependent on each individual action taken. By selecting a greater value, the model gives long-term cumulative rewards higher priority than short-term rewards. Selecting a high γ is optimal in this scenario since optimizing the long-term hit ratio is crucial.

9.1.9 Applicability of reinforcement learning:

Reinforcement learning is a nearly ideal solution for the cache replacement policy problem. First, as discussed in the previous chapter, the issue can be represented as an MDP. Cache replacement policy's MDP is intended to be partially known. This suggests that the majority of the MDP is known, but the transition function P(s'|s,a) is stochastic and not entirely known. Direct optimization approaches cannot be used due to unknown variables. Learning a policy requires regular interactions with the environment. Reinforcement learning is specifically developed to solve this type of difficulty.

Second, the cache replacement policy can be simulated and executed multiple times. Current state-of-the-art reinforcement learning methods remain relatively sample inefficient. This means that they require viewing interacting with an environment a huge number of times in order to achieve appropriate policies. Current algorithms are only effective after millions of games [36]. For the cache problem, data is accessed millions of times per day via caches or

databases across the internet. There are numerous data access traces with hundreds of requests available to simulate cache issues. It is even possible to create purely synthetic data that mimics the access patterns of real data. Reinforcement learning algorithms can interact with the cache environment and effectively learn policies. This capacity to run many training steps enables state-of-the-art reinforcement learning algorithms to be used in this setting.

9.1.10 Components of reinforcement learning in the context of NDN

Agent:

The agent is an intelligent decision-making entity that is often implemented as a reinforcement learning (RL) model that is in charge of controlling a network node's cache or router. Its primary job is to observe the current state of the cache and network environment, choose actions (such as which content to cache or evict), and learn over time to optimize cache performance depending on feedback from the environment [41].

Environment (E):

The environment depicts the NDN network cache system in which the RL agent operates. It contains cache storage, incoming content requests, and network state. The environment responds to the agent's caching and replacement activities by modifying the cache state and giving performance feedback such as reward or penalties.

State(S):

It represents the current content of the cache at any given time.

Action (A):

Regarding cache management, actions are the potential decisions the RL agent may make. This covers prefetching tactics, which content to cache, and which cached content to remove or replace.

Reward(R):

The environment gives the agent a reward signal following each action. This reward measures the action's immediate cost or benefit, such as a decrease in retrieval latency, an increase in cache hit ratio, or a savings in network bandwidth. The agent's objective is to optimize cache performance by gradually increasing the cumulative reward over time.

Policy (π):

A policy defines the agent’s behavior in an environment. The policy tells the agent what action to perform in each state.

- **Deterministic policy :**

tells the agent to perform one particular action in a state ,denoted by **a = π (s)**

- **Stochastic policy:**
- A stochastic policy does not map a state directly to one particular action; instead, it maps the state to a probability distribution over an action space denoted by

$$\pi(\boldsymbol{a} \mid \boldsymbol{s}) = \boldsymbol{P}[\boldsymbol{At} = \boldsymbol{a} \mid \boldsymbol{St} = \boldsymbol{s}].$$

Episode (trajectory)

by (τ), this agent-environment interaction, or in another way the path the agent takes from initial state until the final state called an episode**.**

- **Episodic tasks:** The tasks that has a final (terminal) state.
- **Continuous tasks:** The tasks that doesn't have a terminal state.

Termination condition

Some caching tasks may have predetermined episodes, like after processing a given amount of requests or as a fixed time intervals, which signal the end of an interaction sequence. This enables the agent to assess performance across several episodes and adjust its policy accordingly.

9.1.11 Quality Learning(Q-Learning)

Q-learning is a popular model-free reinforcement learning algorithm based on the Bellman equation. The core concept of Q-Learning is to learn a policy that tells an agent what action to take under what conditions. The algorithm learns a Q-function, known as as $Q(s, a)$, that estimates the total reward an agent can anticipate to get after taking action an in state s and following the optimal policy. The purpose of Q-Learning is to discover the optimal Q-function, $Q * (s, a)$, which reflects the maximum cumulative reward possible from every state-action pair.

9.1.11.1 Value function

The Q value function, denoted $Q(s, a)$, represents the value of a state-action pair. It is defined as the expected cumulative value of future rewards when the agent is in a state s and chooses an action a.

9.1.11.2 Bellman equation

Richard Bellman, a mathematician, devised this equation in 1957 as a method for making optimum decisions via recursion that guides the iterative update of Q-values in Q-learning, balancing current rewards against predicted future rewards in order to learn optimum decision-making strategies over time.

$$\boldsymbol{Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma a' max Q(s',a') - Q(s,a)]}$$

Where:

- $Q(s,a)$ is the current estimate of the Q-value for state-action pair (s,a) .
- r is the reward reicived after taking and action a in $state\ s$.
- s' is the next state after action a .
- $maxa'Q(s',a')$ is the highest Q-value over all possible $actions\ \ a'$ in the next $state\ s'$.
- $\alpha \in (0,1]$ is the learning $rate$.
- $\gamma \in [1]$ is the discount factor that determines the value of future rewards in comparison to new rewards.

9.1.12 Q-table

A Q-table or matrix is created while performing the Q-learning. The table follows the state and action pair, $i.e.,[s,a]$, and initializes the values to zero. After each action, the table is updated, and the q-values are stored within the table.

The RL agent uses this Q-table as a reference table to select the best action based on the Q-values.

We can imagine Q-table as a memory of what the agent learned from experience

9.1.13 Structure of Q-table:

- Rows represent $states\ (S)$.
- Columns represent all possible $action(A)$.
- Each entry in the table represents the Q-value for a state-action pair.

The update of Q-values is done using a mathematical equation that takes into account the current Q value, the immediate reward received, and the maximum Q value for the next state. This equation iteratively refines the Q values throughout the learning process, helping the agent make more informed decisions over time. Essentially, the update aims to balance immediate rewards with expected future rewards, guiding the agent in learning optimal strategies for navigating their environment.

9.1.14 Q-Learning Algorithm

9.1.14.1 Model the cache replacement as a Markov decision process (MDP):

- **State:** Represents the current cache state, including which contents are cached, their popularity, age, retrieval time, and other pertinent features.

• **Action:** $A = \{a_1, a_2, \ldots, a_n\}$ When new content comes and the cache is full, the action is to decide which cached items to be replaced. Alternatively, in certain configurations: whether to cache the new item or not

• **Reward:** After taking an action (eviction), the system receives a reward, typically based on cache hit/miss, retrieval time, or network efficiency. A common reward is +1 for a cache hit and 0 (or negative) for a miss

• **Episodes:** A sequence of actions that ends when the agent reaches a terminal state**.**

9.1.14.2 Initialization: the agent starts with initializing of Q table, where Q-values $Q(s, a)$ are typically initialized to zero.

If state space is way too long we use (DQN).

9.1.14.3 Training Loop:

For each new request:

- Observe the current cache state
- Action selection is done by using ε-greedy strategy pick the best-known action (evict item with highest Q-value, $arg\ max a\ Q(s, a)$), (exploitation) with probability $1 - \epsilon$, or a random action with probability ε, to encourage exploration.
- From the cache, replace the selected content with the new one.
- Receive reward.
- Updating the Q value using the Q function formula.
- $s \leftarrow s'$
- End the episode when a terminal state is reached.

9.1.14.4 Convergence to optimal policy:

As the agent interacts with its environment, it improves its Q-values; gradually determining which cache replacement actions maximize long-term cache hit rates or minimizes retrieval latency.

The policy is optimum when the Q-values stabilize and the agent consistently makes the best eviction decisions for each state [59].

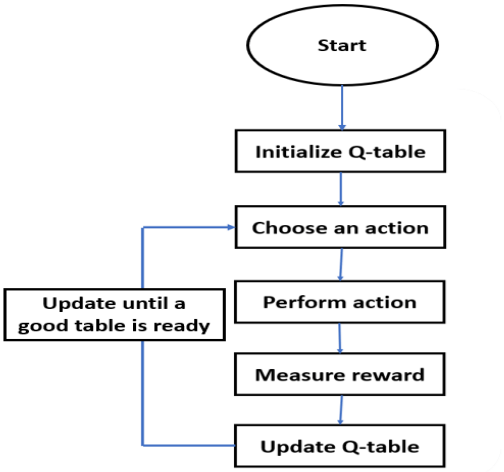

Figure 12 Q-Learning circle

9.1.14.5 Dilemma Exploration vs exploitation:

The exploration-exploitation dilemma (or explore-exploit tradeoff) is a key problem in decision-making and reinforcement learning. It describes the conflict between the two strategies:

9.1.14.6 Exploration: Choosing the best-known option based on current knowledge to maximize immediate reward.

9.1.14.7 Exploitation: Trying new or less-known options to gather more information, which may lead to better long-term outcomes but can result in short-term losses

9.1.14.8 The Exploration-exploitation dilemma: is the challenge of balancing the use of existing knowledge to maximize immediate rewards (exploitation) against the desire to obtain new information which could enhance future decisions (exploration). This tradeoff is fundamental to reinforcement learning and many real-world adaptive systems.

9.1.14.9 ϵ-greedy strategy

An **ϵ-greedy strategy** is commonly used to handle this. With probability ϵ, a random action is chosen (exploration), whereas with probability $1-\epsilon$, the action with the highest Q-value is chosen (exploitation). The value of ϵ often decreases over time, allowing for more exploration at first and more exploitation as the algorithm learns more about the environment.

9.1.14.10 Q-learning Limitation:

Although Q-learning is a powerful reinforcement learning algorithm, it has some drawbacks that prevent it from being effective in increasingly complicated environments.

- **Scalability issues:**

Traditional Q-Learning uses a Q table in which each state-action pair is assigned a Q value. As the state space grows, particularly in continuous or high-dimensional environments, the Q table becomes problematic, resulting in memory inefficiency and a poor learning rate.

- **Discrete state and action spaces:**

Q-Learning performs best in contexts where states and actions are discrete and finite. However, many real-world issues contain continuous state and action spaces, which traditional Q-Learning cannot successfully manage without discretizing these spaces, resulting in knowledge loss and poor strategies.

To address these issues, one alternate strategy is to mix Q-learning and deep neural networks. This approach is known as Deep Q-Learning (DQL). DQL's neural networks approximate the Q value for each pair of states and actions.

9.1.15 Deep Q-Network:

In 2013, Deep Mind published the Deep Q-Network (DQN) method and the article that presents it: "*Playing Atari with Deep Reinforcement Learning*", DQN is designed to learn how to play Atari games. This is a significant advancement in the field of reinforcement learning, paving the door for future improvements in this area. In reinforcement learning, the term "*deep Q-network*" refers to the combination of a deep neural network and the Q-learning method [38].

The neural network receives the input state and generates the Q values for all possible actions. The following figure illustrates the difference between Q-learning and deep Q-learning in the evaluation of the Q-value

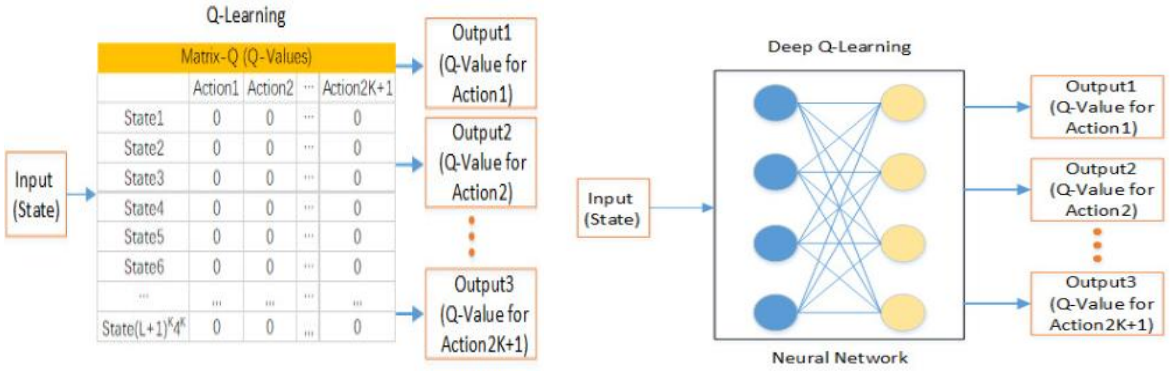

Figure 13 Q-learning and deep Q-learning in the evaluation of the Q value [37].

In reality, this algorithm uses two deep neural networks (DNNs) to stabilize the learning process [39].

- The first is called **the main neural network** , represented by the weight vector θ, and it is used to estimate Q values for the current state s and action a: $Q(s, a;\ \theta)$ in real time.
- The second is **the target neural network**, parameterized by the weight vector θ', and it will have exactly the same architecture as the main network, but it will be used to estimate the Q values of the next state and action.

All learning takes place in the main network. The target network remains frozen (its parameters remain unchanged) for a few iterations, and then the weights of the main network are copied to the target network, thus transferring the knowledge learned from one to the other. This makes the estimates produced by the target network more accurate after copying.

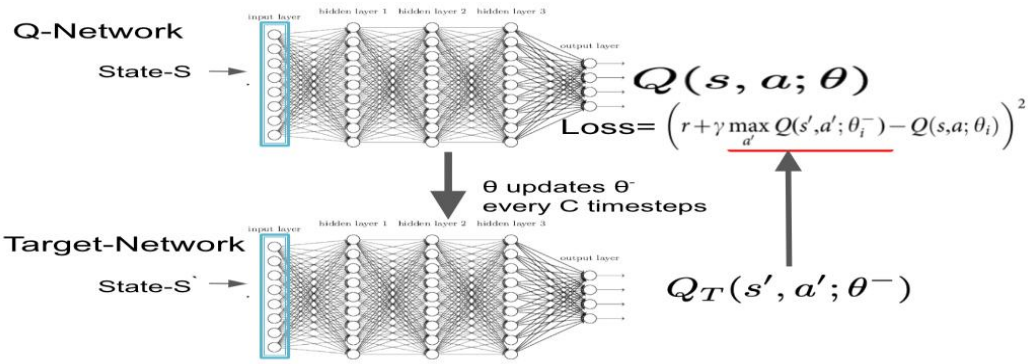

Figure 14 main network and target network

9.1.15.1 Bellman's equation and the loss function for the DQN algorithm:

9.1.15.2 The equation for updating the Q value in the main network:

$$\mathbf{Q(s,a;\ \theta) \leftarrow Q(s,a;\ \theta)\ +\ \alpha\,(r\ +\ \gamma\ max a' Q(s',a';\ \theta')\ -\ Q(s,a;\ \theta\ ))}$$

Where:

- $Q(s, a;\ \theta)$ Represents the estimated Q value for state "s", action "a", and neural network parameters "θ"..
- r is the reward reicived after taking and action a in $state\ s$.
- $maxa'Q(s', a';\ \theta')$ Represents the maximum expected Q value for the next state " s' " and all possible actions " a' ", estimated by the target network with the parameters " θ' ".
- $\alpha \in (0,1]$ is the learning rate .
- γ is the discount factor that determines the value of future rewards in comparison to now rewards.

- $Q(s, a; \theta)$ Represents the current Q-value of the current action-state, estimated by the main network with the "θ" parameters.

9.1.15.3 Loss function

In order to train a neural network, we need a loss (or cost) function, which is defined as the squared difference between the two sides of the Bellman equation, in the case of the DQN algorithm:

$$\boldsymbol{L}(\boldsymbol{\theta}) = [(\boldsymbol{r} + \boldsymbol{\gamma}\, \boldsymbol{max a'} \boldsymbol{Q}(\boldsymbol{s'}, \boldsymbol{a'}; \boldsymbol{\theta'}) - \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}))^2]$$

This is the function that we will minimize using gradient descent, which can be computed automatically using a deep learning library such as TensorFlow or PyTorch. Then the function of updating the weights:

$$\boldsymbol{\nabla_\theta L(\theta)} \leftarrow [(\boldsymbol{r} + \boldsymbol{\gamma}\, \boldsymbol{max a'} \boldsymbol{Q}(\boldsymbol{s'}, \boldsymbol{a'}; \boldsymbol{\theta'}) - \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}))\, \boldsymbol{\nabla_\theta}\, \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta})]$$

9.1.15.4 Target Network:

The target network is a copy of the Q network, which is used to approximate the Q function**.**

The target network maintains a separate weight vector, the target network weights are not updated with each iteration. Instead, they are copied periodically from the evaluation network, creating a time lag between the two networks.

Choosing a separate target network makes divergence unlikely because it adds a delay between when the primary network Q value is updated and when the target Q values are updated. This means that the target network uses the same weights to estimate the target values for a certain period of time, often referred to as a freeze period.

The target network plays a crucial role in providing stable target Q values to guide the training of the main network.

9.1.15.5 Experience replay:

Multiple experiences are collected and stored in a replay buffer within the DQN. The deep neural network is updated using a random sampling of the experiences in the buffer [40]. After a certain number of episodes, a random sampling of experiences (batch) from the replay buffer is used to update the current parameters. After a set number of prediction steps, the prediction network's parameters are copied to the target network.

The DQN agent uses a replay buffer to store past experiences. Each experience is a tuple (state, action, reward, next state) see in (figure) representing a unique transition from one state to another. Replay memory stores these experiences for later sampling by providing a diverse data source and allowing the agent to learn from past experiences repeatedly and efficiency

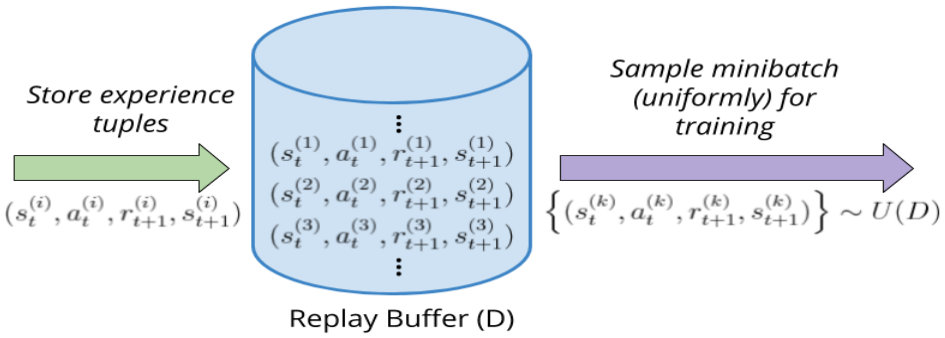

Figure 15 Replay buffer

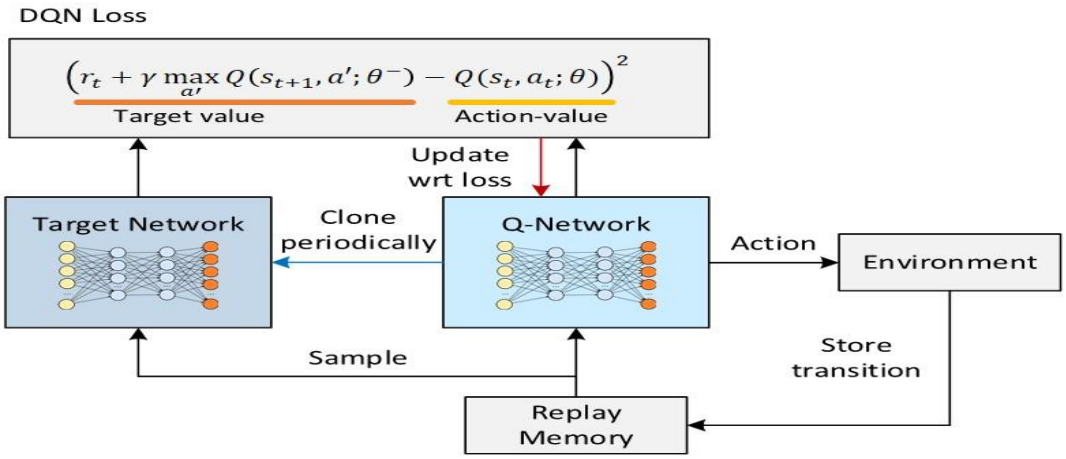

Figure 16 A data flow for a DQN with a replay buffer and a target network [42]

9.1.16 Double Deep Q-Learning:

A key drawback of the DQN algorithm is its tendency to overestimate Q-values, causing the agent to expect higher rewards than it will really receive. This overestimation happens because the online network is employed for both action selection and evaluation, and the update equation is based on a possibly overestimated maximum Q-value. [43]

To address this, Double DQN utilizes two networks: one to choose the action and another to evaluate it, considerably lowering overestimation bias and increasing value estimation accuracy.

9.1.16.1 The principle of the Double DQN algorithm:

The main neural network decides which of all the next best actions is available, and then the target neural network evaluates that action to find out its Q value. This technique solves the problem of overestimation in DQN

In a simple way the main network picks the best action, but does not take its Q value. Instead the target network through that action selection its picks a Q value.

9.1.17 Difference between DQN and DDQN:

9.1.17.1 DQN(Deep Q-Network):

- The update of the Q function in the Deep Q-learning network is based on Bellman's equation for Q values:

$$\boldsymbol{Q(s,a;\,\theta)} \leftarrow \boldsymbol{Q(s,a;\,\theta)} + \boldsymbol{\alpha}\,(\boldsymbol{r} + \boldsymbol{\gamma}\,\boldsymbol{max\_a'}\,\boldsymbol{Q(s',a';\,\theta')} - \boldsymbol{Q(s,a;\,\theta)})$$

- Updating the network weights in the DQN is done by minimizing a loss function that measures the difference between the predicted Q values and the target values. The target values are calculated using Bellman's equation:

$$\boldsymbol{Y} = (\boldsymbol{r} + \boldsymbol{\gamma}\ \boldsymbol{maxa'}\,\boldsymbol{Q(s',a';\,\theta')})$$

And the loss function is:

$$\boldsymbol{L}\,(\boldsymbol{\theta}) = [(\boldsymbol{r} + \boldsymbol{\gamma}\,\boldsymbol{maxa'}\boldsymbol{Q(s',a';\,\theta')} - \boldsymbol{Q(s,a;\,\theta)^2})]$$

The gradient update is :

$$\boldsymbol{\nabla_\theta L(\theta)} \leftarrow \boldsymbol{E}[(\boldsymbol{r} + \boldsymbol{\gamma}\,\boldsymbol{maxa'}\boldsymbol{Q(s',a';\,\theta')} - \boldsymbol{Q(s,a;\,\theta)})\,\boldsymbol{\nabla_\theta}\,\boldsymbol{Q(s,a;\,\theta)}]$$

Using stochastic gradient descent (or variants like Adam), the network parameters are updated to minimize this loss:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\alpha}\boldsymbol{\nabla_\theta}\boldsymbol{L(\theta)}$$

9.1.17.2 DDQN: Updating Q values in Double Q-Learning uses two Q networks, denoted

$Q1\ and\ \theta 1$: for main network

$Q2\ and\ \theta 2$: for target network

- The update of the Q values is based on the following equation:

$$\boldsymbol{Q_1(s,a)} \leftarrow \boldsymbol{Q_1(s,a)} + \boldsymbol{\alpha}[\boldsymbol{r} + \boldsymbol{\gamma}\boldsymbol{Q_2(s',arg\ maxa\ Q_1(s',a\,;\theta_1);\theta_2)} - \boldsymbol{Q_1(s,a)}]$$

- The loss function in the DQN double algorithm:

$$\boldsymbol{L}\,(\boldsymbol{\theta}) = \boldsymbol{E}\,[(\boldsymbol{r} + \boldsymbol{\gamma}\,\boldsymbol{Q_2}\,(\boldsymbol{s'},\boldsymbol{arg\ maxa}(\boldsymbol{s'},\boldsymbol{a}\,;\,\boldsymbol{\theta\_1}\,).\boldsymbol{\theta\_2}) - \boldsymbol{Q(s,a;\,\theta)})^2]$$

The value Q for action a in the state s

- the immediate reward r plus the expected future reward, discounted by the discount factor
- $Q_1(s,a;\,\theta_1)$ is the value Q predicted by the main net for action a in the $state\ s$.
- $Q_2(s', arg\ maxa\ Q_1(s',a\,;\theta_1).\theta_2)$ is the target Q value, calculated from the second network and using the maximum action according to the predictions of the first network, in the following $state\ s'$.

- E: represents the mathematical expectation, i.e. the average over a set of experiment samples.

The updates in the DQN and the Double DQN consist of adjusting the neural network weights by minimizing a loss function, but with a key difference in the calculation of the target values, which allows the Double DQN to reduce the overvaluation of the values

Updates to the weights: $\boldsymbol{\theta \leftarrow \theta - \alpha \nabla \theta [Y - Q(s, a; \theta)}$

$$\boldsymbol{\theta \leftarrow \theta - \alpha \nabla \theta [(r + \gamma Q2\,(s', arg\,max_a\, Q1(s', a\,;\, \theta\,1)\,;\, \theta\,2) - (s, a;\, \theta)]}$$

$$\boldsymbol{Y\ is\ Q\ target\ (r + \gamma Q2(s', arg\,max_a\, Q1(s', a\,;\, \theta\,1)\,;\, \theta\,2)}$$

9.1.18 Double DQN Algorithm

This algorithm has been shown to function effectively with huge state spaces, making it suitable for solving cache replacement policies [40].

- **Initialization:**

Main network: This network will learn which cache items to evict, initialized with random weights.

Target network: A copy of the main network, updated periodically.

Replay buffer: a queue to stores past experiences (cache states, actions, rewards, next state).

- **Data collection:**

The agent interacts with the caching environment, which receives requests for data items based on a Zipf distribution.

At each time step, the agent chooses a cache replacement action based on an exploration strategy (epsilon-greedy), observes the reward (cache hit or miss), and determines the new cache state.

The replay memory will store the transition (current cache state, action taken, reward received, and next cache state).

- **Mini batch sampling :**

At regular intervals, the agent selects a random subset of transitions, called mini-batches, from the replay memory to update the core network settings.

- Q target calculation:

For each transition in the mini-batch, calculate the Q target as follows:

a) Uses the mainnet to estimate the value of the optimal action for the new state.

b) Uses the mainnet to select the optimal action for the new state.

c) Uses the target network (a frozen copy of the mainnet) to estimate the value of this action.

- **Loss function calculation**
- **Weight update:**

a) Uses an optimization algorithm such as stochastic gradient descent to update mainnet weights.

b) Periodically updates the target network weights by copying the primary network weights.

- Repeat steps 2 through 6 for a fixed number of iterations or until convergence is reached.
- **Evaluation**

Periodically evaluate the agent's performance by testing the learnt cache replacement policy without exploration (i.e., greedy decisions) to measure cache hit ratio and overall efficiency.

- **Termination:**

Stops training when desired performance is achieved or when the predefined number of iterations is completed.

9.1.19 Conclusion

This chapter gave a full review of cache replacement techniques in Named Data Networking (NDN), starting with standard policies and progressing to intelligent, learning-based methods. Initially, we looked at traditional caching strategies like LRU and LFU, which, despite their simplicity and minimal computational overhead, struggle to adapt to dynamic and content-centric network environments like NDN.

To address these challenges, we developed intelligent cache replacement methods based on reinforcement learning. The problem was initially formulated as a Markov Decision Process (MDP), which allowed cache decision-making to be modeled as a sequential, state-dependent optimization process. Q-Learning, a foundational method in this domain, provided a simple but effective framework for learning optimum policies through trial-and-error interactions.

Building on this, Deep Q-Networks (DQN) were used to handle enormous state spaces by approximating Q-values with neural networks. DQN enhanced scalability and learning capabilities, but it suffered from Q-value overestimation. This was significantly addressed by Double DQN, which divides action selection and assessment between two networks, resulting in more accurate and stable learning.

Chapter-III-

Modeling and interpretation

1. Introduction:

This chapter discusses the design and implementation of intelligent caching methods in Named Data Networking (NDN) utilizing advanced deep reinforcement learning techniques. Building on the theoretical foundations discussed in the previous chapter, we now move on to the practical implementation of these models, with especially focused on the Dueling Deep Q-Network (Dueling DQN) and the proposed enhanced architecture, which includes Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) units.

The Dueling DQN framework refines the estimate of action values by dividing the Q-function into two different components: the state-value function and the advantage function. This design allows the agent to learn which states are valuable regardless of specific actions, which is especially beneficial in caching situations where certain actions have little effect on the overall system state.

To improve the model's ability to capture complex patterns in data requests, we propose an advanced Dueling DQN architecture that combines CNN layers for extracting spatial characteristics from the content request history with LSTM layers for learning temporal dependencies. This enhaced architecture is intended to dynamically adapt to changing content popularity and user behavior, resulting in more informed and efficient cache replacement decisions.

We describe the system design, which includes the network architecture, input and output representations, and hyperparameter settings. The training process adopts a Zipf-distributed request pattern to simulate realistic content access behavior in NDN. A replay buffer is utilized to save state transitions, while mini-batch sampling is used to provide steady learning.

Finally, we provide the evaluation results, which compare the performance of the proposed enhaced model to baseline techniques such as classic cache replacement strategies and standard reinforcement learning strategies. Key performance characteristics such as cache hit ratio and latency, are examined to verify the proposed model's usefulness in optimizing caching decisions in an NDN environment.

Our approach, detailed in the following section, aligns perfectly with this context. In simpler terms, it is designed based on the aforementioned notion, aiming to further enhance the caching performance and overall improve the network performance.

2. Theoretical background:

2.1 Duel DQN(DDQN)

The Dueling DQN architecture was developed by Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. This architecture was presented in their research paper titled "Dueling Network Architectures for Deep Reinforcement Learning," published in 2016.

The proposed network architecture under the name dueling architecture, This approach distinguishes between state values and state-dependent action advantages. The dueling architecture is made up of two streams that represent the value and advantage functions, whereas sharing convolutional feature learning module [60].

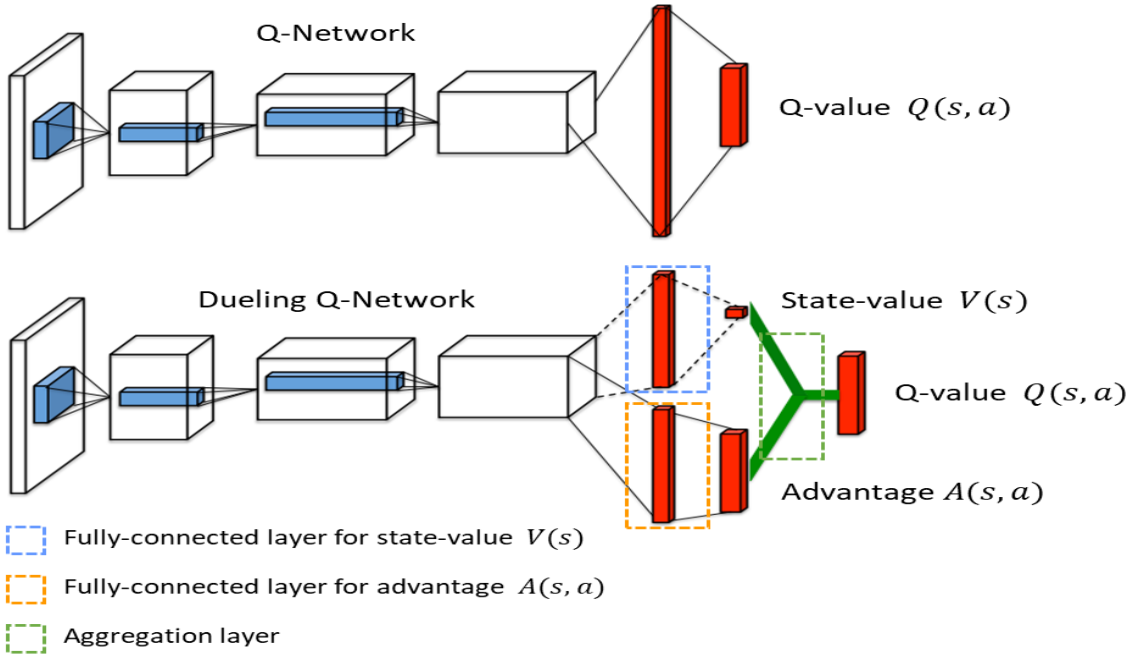

Figure17 A popular single stream Q-network (top) and the dueling Q-network (bottom) The dueling network has two streams.

The two streams are gathered using an aggregating layer to estimate the state-action value function Q, as shown in the Figure above. This dueling network is a single Q network with two streams, replacing the common one-stream Q network in existing algorithms, such as Deep Q-Networks (DQN; Mnih et al., 2015). Without any additional supervision, the dueling network automatically generates separate estimates of the state value function and advantage function.

2.2 Dueling architecture:

The dueling architecture may intuitively learns which states are valuable (or not), without needing to learn the impact for every action for every state. This is especially helpful in states where its actions do not affect the environment in any relevant way [60].

In dueling architecture, the action-value function $Q(s,a)$ is decomposed into two parts:

- The value of state $V(s)$ represents how valuable the $state\ s$ is, regardless of the chosen action.
- The advantage of each action $A(s,a)$ represents how much better the action a is compared to the average of the actions in the $state\ s$.

Similar to the original DQNs, the dueling network's lowest layers are convolutional (Mnih et al., 2015). However, utilize two sequences (or streams) of fully connected layers rather than one series of convolutional layers made up of single layers. The streams are designed in a way that allows them to provide distinct estimates of the value and advantage functions. Ultimately, a single output function is generated by combining the two streams.

As in (Mnih et al., 2015), the output of the network is a set of Q values, one for each action.

The dueling network can be trained using any of the numerous known techniques, including DDQN or Q-learning , because its output is a Q function. Furthermore, it can take advantage of any enhancements to these algorithms, such as enhanced intrinsic motivation, better exploration rules, replay memories, and so forth.

Very careful design is needed for the module that outputs a Q estimate by combining the two streams of completely connected layers.

From the expressions for advantage $Q(s,a) = V(s) + A(s,a)$ and state value $V(s) = E(s)[Q(s,a)]$ it follows that $E[A(s,a)] = 0$. Moreover, for a deterministic policy, $a* = argmaxa\ Q(s,a')$, it follows that $Q(s,a) = V(s)$ and hence $A(s,a) = 0$.

In previous figure, we have a dueling network with two streams of completely connected layers. One stream produces a $scalar\ V(s;\theta,\beta)$, while the other stream produces $a\ |A|$ dimensional $vector\ A(s,a;\ \theta,\alpha)$. Here, θ stands for the convolutional layer parameters, and α and β for the two streams of fully-connected layers.

We could be tempted to build the aggregating module in the manner shown below using the concept of advantage:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha)$$

Keep in mind that this expression is applicable to every occurrence of (s,a); in other words, we must replicate the scalar, $(s;\theta,\beta)$, $|A|$ times in order to represent the equation above in matrix form.

We must remember that $Q(s,a;\theta,\alpha,\beta)$ is only a parameterized estimate of the true Q-function. Furthermore, it would be wrong to claim that $A(s,a;\theta,\alpha)$ offers a reasonable estimate of the advantage function or that $V(s;\theta,\beta)$ is a good estimator of the state-value function.

The equation above is unidentifiable in the sense that given Q, we cannot retrieve V and A uniquely. To demonstrate, add a constant to $V(s;\theta,\beta)$ and subtract the same constant from$(s,a;\theta,\alpha)$. This constant cancels out, giving in the same Q value. This lack of identifiability is reflected in poor practical performance when using this equation directly.

To overcome the issue of identifiability, we can make the advantage function estimator have no (zero)advantage at the chosen action. That is, we let the last module of the network implement the forward mapping.

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - argmax\_a' A(s,a';\theta,\alpha))$$

For

$$\boldsymbol{a* = argmaxa\, AQ(s,a';\theta,\alpha,\beta) = argmax\, A(s,a;\theta,\alpha)},$$

we get

$$\boldsymbol{Q(s,a*;\theta,\alpha,\beta) = V(s;\theta,\beta).}$$

The stream $V(s;\theta,\beta)$ provides an estimates the value function, whereas the other stream produce an estimates the advantage function.

An alternative module replaces the max operator with an average:

$$\boldsymbol{Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum A(s,a';\theta,\alpha))}$$

On the one hand, this loses the original semantics of V and A because they are now off-target by a constant, but on the other hand, it improves the optimization's stability: with (9) the advantages only need to change as fast as the mean, rather than having to compensate for any change in the optimal action's advantage in (8). Also tested a $softmax$ version of equation (8), but it produced identical results as the simpler module of equation (9).

Note that, while subtracting the mean in equation (9) improves identifiability, it has no effect on the relative rank of the A (and hence Q) values, keeping any greedy or greedy policy based on Q values from equation (7). When acting, it is sufficient to evaluate the advantage stream before making decisions.

It is critical to note that equation (9) is seen and implemented as part of the network, rather than as a separate algorithmic step. Dueling designs, like typical Q networks (e.g., Mnih et al.'s deep Q-network), may be trained using simply back propagation. The estimations $V(s;\theta,\beta)$ and $A(s,a;\theta,\alpha)$ are computed automatically, with no further supervision or algorithmic modifications.

2.3 Convolutional Neural Network (CNN):

CNN is a feed forward neural network (FNN) that includes an input layer, an output layer, and hidden layer, these hidden layers are represented by convolutional layers combined with pooling layer. The primary concept behind CNNs for numeric data involves applying convolution to local temporal windows of the input data, which enables the network to capture temporal patterns and dependencies within the data. CNNs perform well in recognizing capturing both local and global patterns in time series data, making them suitable for various like time series forecasting, anomaly detection, and other signal processing tasks. Even with their popularity and proved efficiency, CNNs designed for numeric data still struggle with issues such as missing data, as well as long-term dependencies. The development of more resilient and effective CNN designs has been the focus of recent research advancements. Examples of these include the WaveNet and Temporal Convolutional Network (TCN) models, it has produced encouraging results in a variety of applications.beside numerical data CNNs have been used to several data formats, including text and graphs, displaying their versatility and adaptability and promise to boost deep learning [63, 65].

CNNs are a strong tool for processing many types of data, and their continuing development and optimization have the potential to enhance the area of artificial intelligence and its applications in diverse domains [37].

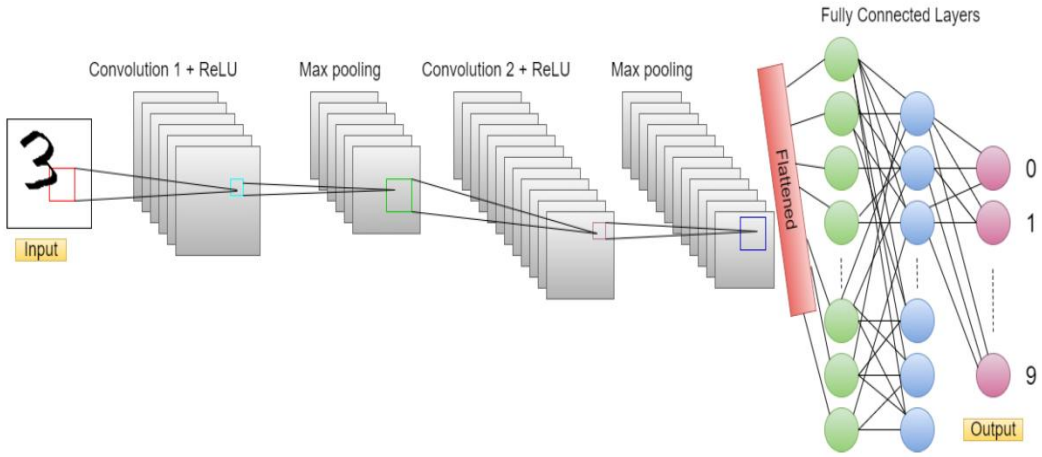

Figure 18 convolutional neural network (CNN)

A typical CNN will include the following layers: input, convolution, ReLU, pooling, flattening and fully connected layer.

- **Input:** the input layer is where the data is fed into the network, each is represented as a grid of pixel values, and this layer is responsible for passing this information to the subsequent layers. Its represented as a matrix of pixels.
- **Convolutional:** convolutional layers scan the input data using filters (kernels) known as a feature detector to detect patterns, It requires a few components, which are input data, a filter and a feature map. These layers utilize filters to smooth input signals and create feature maps for a dataset. These maps are activated using convolution with a kernel over the dataset.
- **Relu(Activation function-using layer):** a layer that uses the previous layer's output to activate its own output, also known as a rectified linear unit layer. RELU adds non-linearity to the network in a unique way.
- **Polling:** also known as down sampling, does dimensionality reduction, which reduces the amount of parameters in the input. The pooling process, like the convolutional layer, sweeps a filter across the whole input; however, this filter has no weights. Instead, the kernel uses an aggregation function on the values in the receptive field to populate the output array. While the pooling layer loses a lot of information, it also provides some benefits to the CNN. They assist to minimize complexity, increase efficiency, and lessen the danger of overfitting.

- **Flattening:** Once the feature has been extracted, the data is converted into a vector and passed through fully connected layers for classification.

- **Fully connected layer:** This layer may be called the "output" layer, it provides the final prediction using a Softmax function for classification tasks.

2.4 Long Short Term Memory (LSTM):

LSTM, a form of recurrent neural networks (RNN), is a popular deep learning technique [65]. have many interconnected layers, although interactions between the four levels differ. The LSTM model has memory cells, which are controlled by gates. There are three different sorts of entry gates (input gate, output gate, and forget gate). These gates are used to alter data in an LSTM, a fixed amount of training data can be saved in the memory module. Cell state memory enables LSTM to recollect long-term dependencies [63]. Cell state memory is the memory unit that enables LSTM to recall long-term dependencies. There are three major types of gates: the forget gate, the input gate, and the output gate [66].

The memory cell remembers values across various time periods. The three gates accept and reject information that passes through the cell. The forget gate in Figure determines which information will be remembered from the previous cell state (Ct−1). This choice is made using the sigmoid activation function (σ). This sigmoid's output is f(t). If the output value is 1, the data is entered into the model; otherwise, if its 0, the data will not be passed through it this sigmoid's input is the current input (xt) and the prior hidden state (ht-1). The input gate decides what novel information will be stored in the current cell state Ct [64].

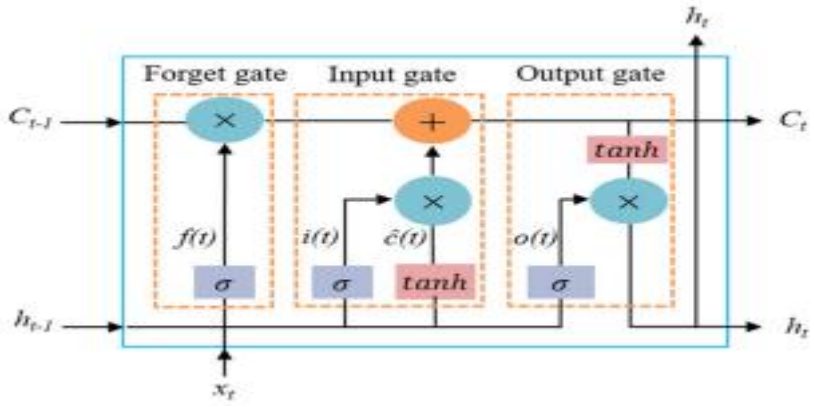

Figure 19 long short term memory (LSTM) [64]

- Hidden State An LSTM layer's output, known as the hidden state, is utilized as input in the next layer. To indicate how much of each piece should be sent, the sigmoid layer produces values between 0 and 1. The Tanh layer generates new state-enhancing vectors [63].

$$h_t = o_t \cdot tanh(c_t)$$

• Forget gate: The information that is no longer useful in the cell state is removed with the forget gate. Two inputs x_t (input at the particular time) and h_t-1 (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state, the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.

$$f_t = \sigma\left(W_f \cdot [h_t - 1, x_t] + b_f\right)$$

• Input gate: Decides which new information should be added to the cell state.

$$i_t = \sigma\left(W_i \cdot [h_t - 1, x_T] + b_I\right)$$

• Output gate: Determines what information from the current cell state should be outputted.

$$o_T = \sigma\left(W_O \cdot [h_T - 1, x_T] + b_O\right)$$

2.5 Proposed Model

This part provides a detailed description of the proposed model and algorithm, as well as a discussion of the research process that led to its development.

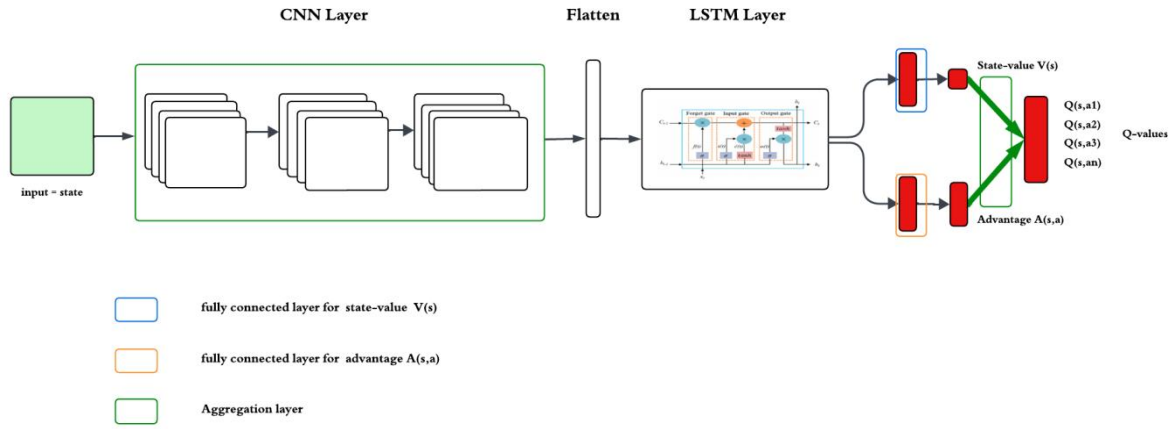

Figure 20 the proposed advanced duel DQN MODEL Architecture

2.6 Problem identification:

While Dueling Deep Q-Networks (Dueling DQN) have been shown to improve the stability and performance of reinforcement learning agents by decoupling the estimation of state values and action advantages, this architecture has significant limitations when used for cache replacement in dynamic networking environments such as Named Data Networking (NDN). Specifically, the Dueling DQN architecture lacks the capacity to detect temporal

relationships in content request patterns, which is critical for making informed caching decisions.

In real-world situations, user requests do not occur at random but rather follow temporal trends—some content becomes popular for short periods of time (temporal locality), while others stay commonly requested throughout time. The Dueling DQN, which processes each state individually, inherently lacks memory and cannot distinguish time-dependent behaviors. As a result, it may struggle to generalize across changing access patterns and cannot fully exploit the sequential structure of cache request streams.

To address these challenges, we add long-short term memory (LSTM) layer to the designed architecture, the LSTM is especially added to represent long-term temporal correlations in in sequential data. By integrating LSTM with the Dueling DQN, the model has the capacity to remember and learn from previous request patterns, which improves its knowledge of when certain content is likely to be reused. This temporal modeling capacity improves the agent's policy, allowing it to make more precise and adaptable cache replacement decisions.

The resulting hybrid model, CNN-LSTM + Dueling DQN, uses CNN for local feature extraction (e.g., spatial patterns in content features) and LSTM for temporal pattern recognition, making it more suited to the complex, time-varying needs of real-world NDN caching systems.

2.7 research methodology

As noted the proposed hybrid architecture aims to improve the performance of duel DQN that removes content based on certain features from the cache when it rich the fill up point in order to make room for the new data. This architecture leverage the strengths of both the convolutional neural network (CNN) and long-short term memory (LSTM) while combined within the duel architecture to address the challenge of cache replacement in named data networking (NDN). This model is designed to capture both spatial and temporal dependencies in content request patterns and make a solid and adaptive replacement decisions in dynamic environment. Content request sequences are generated according to a Zipf distribution as a reference model to represent the pattern of the consumer’s requests [68].

2.8 Zipf distribution

Many studies from past year suggests that Zipf's distribution accurately represent the request frequency of contents, in several scenarios such as web,video on demand (VoD), and user generate content (UGC) in intermediate routers, in terms of cache dimensions and hit ration. ISPs employ caching to swiftly provide the users' request for web content by copying and storing frequently requested files "near" consumers on the network. However, the success of caching is strongly reliant on Zipf's law [68].

Zipf's law says that the frequency of a word in a corpus of natural language utterances is inversely proportional to its rank in the frequency table (i.e., the smaller the rank, the greater the request frequency).so the most frequent word will appear approximately twice as often as the second most frequent word, and three times as often as the third most frequent word. For example in the brown corpus of American English text, the word "The" is the most common(7% of all words), followed by "of" (3,5%) and then "and"(2.8%), etc [40]. Zipf's parameter, α, has a significant impact on how well the network caches content. The probability of requesting the content with rank i can be written as follows, assuming that M represents the content catalog cordiality and $1 \leq i \leq M$ represents the rank of the i-th most popular content.

The probability of requesting the content with rank i expressed as:

$$\boldsymbol{P(X = i) = \frac{1/i^{a}}{\sum 1/j^{a}} = \frac{1/i^{a}}{C}}$$

- M the total number of unique contents (catalog size).

With $\boldsymbol{C = \sum 1/j^{a}}$, the normalization constant to make sure the total probability sums to 1. The skewness of content requests, which is controlled by α, has a significant impact on how well caching methods work. User requests become more focused on the most popular contents as α rises. For instance, when α = 1.2, around 2500 products in a catalog of 100,000 items represent 95% of all requests. However, this quantity drastically decreases to barely 700 items as α rises to 1.4. This shows that caching efficacy is significantly impacted by even small changes in α. A larger α indicates a more skewed distribution, meaning that less content is required in cache to fulfill most user requests. As a result, Zipf's law provides a basis for modeling and assessing cache replacement tactics in contemporary content-centric networks in addition to supporting the need for caching in intermediate routers [71].

2.9 Hyperparameter tuning

In the implementation of the proposed CNN-LSTM integrated with the duel architecture, crucial hyper parameters were carefuly selected and tuned to ensure an optimal performance in the context of intelligent cache replacement in named data networking (NDN),the learning rate was set to 0,001, allowing the optimizer to update weights with a balanced pace of convergence. We used the **Adam optimizer**, known for its adaptive learning capabilities. A batch size of 64. the model was trained over 1000 episode, the zipf distribution is chosen to be 1, The batch size for each update is 10.

Three convolutional layers are used with 32, 64and 128 filters in sequence, utilizing kernel 3x3, which captures local spatial correlations among content request features. Also, the LSTM component has 1units with 2 hidden layers, allowing it to capture longer temporal dependencies in the request sequences. A 0, 2 dropout rate was used after the LSTM layer to overfitting, by randomly turning off some of the neurons during training. In this case, the other the ReLU activation function was used in the CNN layers for non-linearity as well as for minimizing the residual gradient flow to the CNN layers while, in the LSTM, tanh was used to output sequential dependencies within bounded outputs. The outputted Q values were calculated through the dueling architecture which separates the estimate of state value from the action-advantage estimations.

A buffer of size 10000 was utilized to stabilize learning and enhance generalization, enabling the agent to learn from previously stored transitions drawn from random samples. A target network was updated at a slower rate to decouple the target Q-value computation from the online learning updates. An epsilon greedy exploration-exploitation mechanism was employed for managing the trade-off, where it decayed from 1 to 0, 01 throughout the period.

All hyper-parameters were experimentally fine-tuned using validation-based performance analysis, with a focus on cache hit ratio, convergence stability, and latency reduction. This tuning approach considerably improved the model's capacity to respond to dynamic content request patterns while also increasing cache efficiency.

2.10 Proposed dueling DQN Model

In the research we present an intelligent cache replacement strategy for Named data networking (NDN), the strategy is based on a Dueling Deep Q-network where content requests are generated using Zipf distribution to accurately reflect real-world content popularity in which a small subset of content is requested more frequently than the rest. Our model processes recent request histories by encoding each item as a one-hot vector and

stacking the latest requests into a matrix. This matrix is fed into three 1D Convolutional Neural Network (CNN) layers, where convolutional filters slide along the item dimension to extract local patterns—such as repeated requests or frequent item co-occurrences. The feature maps from each layer undergo ReLU activation and max pooling to highlight key features and reducing dimensionality. The CNN stack collects abstract spatial characteristics, resulting in compressed key access patterns.

The spatially compressed output is reshaped and fed into an LSTM network, which processes the CNN-derived vectors one step at a time. At each step, the LSTM changes its hidden and cell states using gating mechanisms that regulate the flow of incoming input, retained memory, and output. For instance, the CNN can recall a rapid rise in demand for a certain item by changing the internal state of the LSTM, while discarding irrelevant input. As the sequence proceeds, the LSTM learns about short-term variations and long-term patterns, such as progressive rises in item popularity or periodic demand cycles.

The LSTM's final hidden state captures the temporal dynamics of the whole sequence, providing a high-level summary for decision-making. This representation is sent into the DQN's dueling streams, which individually assess the state value and advantage of a prospective action. These components are combined to provide Q-values that inform cache replacement decisions. This architecture separates between value and advantage stream, where Stability and robustness are enhanced, especially when various actions provide comparable results. By utilizing reinforcement learning in conjunction with an experience replay buffer, which stores previous transitions, and enables training on diverse, uncorrelated samples.

Our system learns to dynamically modify its cache replacement strategy by combining CNN-based spatial pattern recognition, LSTM-based temporal learning, and the battling DQN architecture. It is particularly well-suited for content-centric networks such as NDN as it efficiently optimizes cache hit rates in non-uniform and time-varying request contexts.

2.11 Experimental Results and Analysis

Here,

2.11.1 Evaluation Metrics

Evaluation measures are used to assess NDN-based caching and networking performance [72]. According to the relevant research, the most essential measures have been established and utilized to assess the quality of caching performance. This work considers the following performance measures:

- **Cache hit ratio (content hit ratio**): The core parameter for evaluating NDN cache performance is the rate of requests fulfilled by all caches in the network that store content locally for a set amount of time [73,74].

$$C_{Hit\_Ratio} = \frac{\boldsymbol{total\ number\ of\ cache\ hits}}{\boldsymbol{total\ interest}}$$

- **Latency:** Latency refers to the overall time it takes to process a content request and return the related data to the user. It is an important performance metric for network systems. Lower latency means faster content delivery, which enhances the user experience dramatically.
- **Average delay (A_Delay):** Is the average time it takes for a consumer to obtain the content after submitting a related interest [52].

$$A_{Delay} = \frac{global\ average\ delay}{number\ of\ consumers}$$

- **Network traffic (Net-traffic):** This represent the total number of interest and data packets received across all routers [52].

$$Net\_Traffic = (interests + data)\ received\ by\ all\ routers$$

2.11.2 Results

This section provides experimental outcomes of our suggested deep reinforcement learning-based cache replacement approach in the context of Named Data Networking (NDN). Our main goal is to assess the suggested model's performance and adaptability, where Key performance metrics such cache hit ratio, average latency, network traffic and producers' load are the main focus of evaluation. To demonstrate the benefits of our suggested approach, the outcomes are examined in relation to traditional cache replacement techniques.

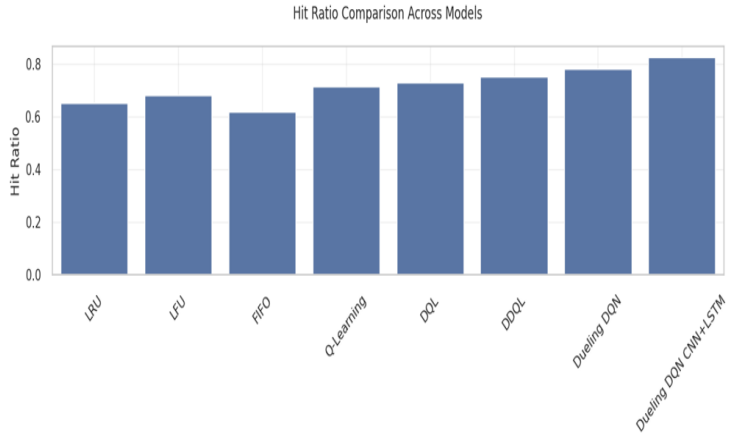

Figure 21 Hit Ratio Comparison across Models

The bar chart compares cache hit ratios produced by various cache replacement algorithms. The x-axis displays the tested algorithms, which include classical approaches like LRU (Least Recently Used), LFU (Least Frequently Used), and FIFO (First-In-First-Out), as well as reinforcement learning-based techniques like Q-Learning, DQL (Deep Q-Learning), DDQL (Double Deep Q-Learning), Dueling DQN, and Dueling DQN CNN+LSTM. The y-axis represents the hit ratio, which is the percentage of cache requests successfully supplied from the cache.

The results indicate that traditional policies like LRU, LFU, and FIFO have moderate hit rates, ranging from 0.61 to 0.67. Reinforcement learning-based strategies have a distinct edge, with Q-Learning obtaining a hit ratio of 0.70, followed by Deep Q-Learning (DQL) and Double Deep Q-Learning (DDQL) at around 0.72 and 0.74, respectively. Notably, the Dueling DQN architecture and its upgraded form, which combines CNN and LSTM processes, have the greatest hit ratios, topping 0.78 and nearing 0.80. This demonstrates the models' capacity to learn and generalize effective cache replacement techniques, resulting in improved content delivery performance.

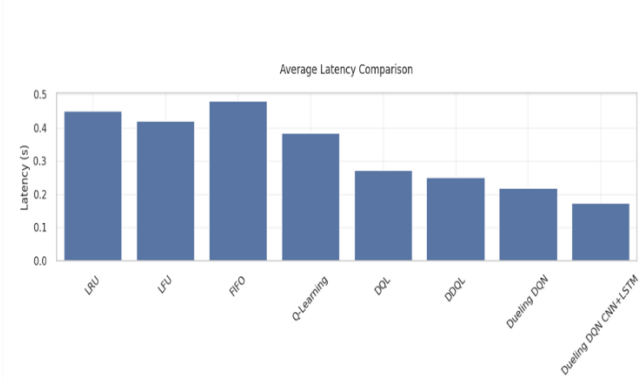

Figure 22 Average Latency Comparison of Cache Replacement Policies

The bar chart "Average Latency Comparison" depicts the average response latency for each cache replacement strategy (in seconds). The x-axis displays the assessed techniques, which include both classical (LRU, LFU, FIFO) and reinforcement learning-based algorithms (Q-Learning, DQL, DDQL, Dueling DQN, Dueling DQN CNN+LSTM). The y-axis shows the average latency in seconds.

As seen, traditional policies such as Least Recently Used (LRU), Least Frequently Used (LFU), and First-In-First-Out (FIFO) have greater latency values, with FIFO reaching roughly 0.48 seconds. In contrast, reinforcement learning approaches show significant progress in latency reduction. Q-Learning has a latency of around 0.38 seconds, although Deep Q-Learning (DQL), Double Deep Q-Learning (DDQL), and Dueling DQN all improve performance. Notably, the Dueling DQN coupled with a CNN-LSTM architecture has the lowest latency, at 0.17 seconds, demonstrating the efficiency of deep reinforcement learning combined with temporal and spatial feature extraction in reducing network response time.

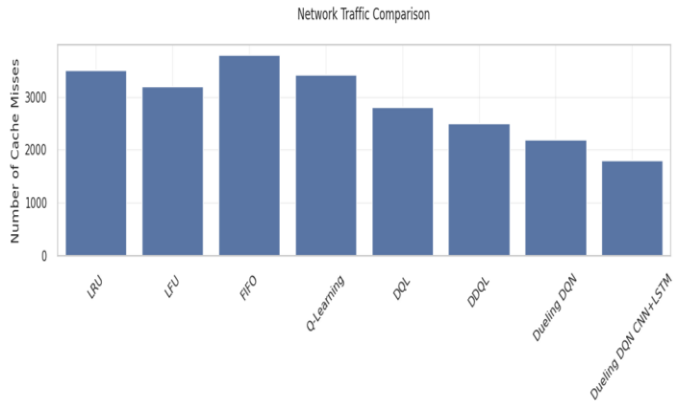

Figure 23 Network Traffic Comparaison between tradition policies and RL approaches

The bar chart named compares the amount of cache misses recorded by different cache replacement algorithms and reinforcement learning-based approaches. The x-axis shows the evaluated algorithms, which include traditional methods like LRU (Least Recently Used), LFU (Least Frequently Used), and FIFO (First-In-First-Out), as well as advanced reinforcement learning techniques like Q-Learning, DQL (Deep Q-Learning), DDQL (Double Deep Q-Learning), Dueling DQN, and Dueling DQN CNN+LSTM. The y-axis shows the total number of cache misses detected for each approach.

The results show that traditional algorithms (LRU, LFU, and FIFO) have a larger amount of cache misses, with FIFO doing the worst. of contrast, reinforcement learning-based approaches show a steady reduction of cache misses, with the Dueling DQN CNN+LSTM

approach obtaining the lowest value. This trend demonstrates the higher effectiveness of deep reinforcement learning models in improving cache management and reducing network traffic. Overall, the figure clearly shows the performance gap between traditional and learning-based policies, highlighting the potential of sophisticated neural architectures to improve cache replacement policies in network contexts. This graphic demonstrates the efficacy of incorporating deep learning techniques to significantly increase network resource consumption.

TABLE: Detailed Comparison Results

Model	Hit-Ratio	Avg-Latency	Network-Traffic	Time Execution
LRU	0.6500	0.4500	3500	4503.00s
LFU	0.6800	0.4200	3200	7451.00s
FIFO	0.6200	0.4800	3800	3009.00s
Q-Learning	0.7150	0.3850	3423	7890.00s
DQL	0.7287	0.2713	2800	47001.00s
DDQL	0.7501	0.2499	2500	67009.00s
Dueling DQN	0.7809	0.2191	2200	950767.00s
Dueling DQN CNN+LSTM	0.8256	0.1744	1800	1804000.00s

Conclusion

This chapter presents a hybrid deep reinforcement learning model that combines CNN, LSTM, and the Dueling Deep Q-Network (Dueling DQN) architecture. The model was created to solve constraints in traditional caching methods by simultaneously capturing geographical and temporal correlations in content request patterns.

Extensive simulations using Zipf-distributed request sequences showed that the proposed CNN-LSTM Dueling DQN architecture outperformed both classic policies (LRU, LFU, FIFO) and standard reinforcement learning models (Q-learning, DQN, DDQN). Evaluation parameters such as cache hit ratio, latency, and network traffic show that the hybrid architecture significantly improves efficiency and flexibility under dynamic request settings.

These results support the efficiency of mixing spatial and temporal deep learning approaches inside reinforcement learning frameworks to control caching in content-centric networks. Future research might concentrate on real-world installations, scaling to bigger topologies, and using content popularity prediction algorithms to increase caching performance even more.

General Conclusion

1. General conclusion:

The NDN concept represents a dramatic shift in network design, altering how data is transmitted and retrieved. Unlike typical IP-based networks, NDN focuses on content, making information distribution more efficient and secure.

Caching plays a critical role in named data networking (NDN) routers, where it stores temporarily copies of data, reducing the need to retrieve it from the original source. However due to the limited cache size, a robust cache replacement policies are important for decision making about which data to remove from the cache when it reaches the fill up point.

In this work, reinforcement learning is employed to try to enhance cache replacement policies. The problem of cache replacement policy is given as a partially known Markov decision process. Recent state-of-the-art-deep reinforcement learning were investigated, several approaches were used including value-based algorithms starting with Q-learning, then the integration of deep learning with Q-learning (DQN), and double deep Q-learning.

In addition to reinforcement learning, an advanced model was used as a novel approach to improve standard dueling DQN algorithm. The algorithm is supplemented with CNN and LSTM. CNNs help the model uncover spatial connections in content request distributions, whereas LSTMs capture temporal relationships across time. Implemented and evaluated with the baseline policies, these two factors appear to suggest that the method provided here might yield even better outcomes with more research on more difficult cache problems. The suggested model beats standard techniques in terms of cache hit ratio, latency reduction, and flexibility to dynamic traffic.

This study demonstrates the potential for merging deep learning with NDN to develop smarter, more efficient network systems. The algorithms show greater performance compared to baselines as the problem gets more complex. The model was meant to simulate real world problems, although performance should be examined in real-world scenarios. For future research the methods and techniques discussed here should be expanded upon and investigated in actual database cache systems.

References

[1]Named Data Networking: Motivation & Details.

https://people.eecs.berkeley.edu/~sylvia/cs268-2014/papers/ndn-overview.pdf

[2] NDN Team. NDN Frequently Asked Questions (FAQ).

https://named-data.net/project/faq/ [Online ; accessed 2-mai-2016]

[3] L. Zhang et al., "Named data networking," ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, pp. 66–73, 2014.

[4] Ruijuan Zheng , Bohan Zhang , Xuhui Zhao , Lin Wang , Qingtao: AReceiver-Driven Named Data Networking (NDN) Congestion Control Method Based on Reinforcement Learning.

[5] Kamorudeen Akindele Amuda, Wakili Almustapha, Pranay Tiruveedula , Ciana Hoggard Binkam Deepak. Revolutionizing Networking Paradigms: A Comprehensive Exploration of Information-Centric Networking (ICN), Content-Centric Networking (CCNx) and Named Data Networking (NDN).

[6] Rama Krishna Thelagathoti , Spyridon Mastorakis , Anant Shah , Harkeerat Bedi , Susmit Shannigrahi: Named Data Networking for Content Delivery Network Workflows.

[7] P. K. Shah, "An O (1) algorithm for implementing the LFU cache eviction scheme," no. 1, pp. 1–8, 2010.

[8] B. leiner et al., "a brief history of the internet," computer communication review, vol. 39, pp. 22–31, jan. 2009, doi : 10.1145/1629607.1629613.

[9] J. Roberts, «The clean-slate approach to future internet design: a survey of research initiatives,» Annals of telecommunications, vol. 64, n° %15, pp. 271-276, 2009.

[10]C. N. V. V. A. S. N. F. C. T. X. V. K. V. K. e. G. C. P. G. Xylomenos, «A survey of information-centric networking research,» Communications Surveys & Tutorials, IEEE, vol. 16, n° %12, pp. 1024-1049, 2014.

[11]George Xylomenos, Christopher N. Ververidis, Vasilios A. Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V. Katsaros, and George C. Polyzos. A Survey of Information-Centric Networking Research.

[12]L . Zhang, D. Estrin, J. Burke, V. Jacobson, J.D. Thornton, D.K. Smetters, B. Zhang, G. Tsudik, K.C. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, and E. Yeh. (2010). Named Data Networking (NDN) Project. [Online]. Available: http://named data.net/project/annual-progress-summaries/ L. Zhang, D. Estrin, J. Burke, V. Jacobson.

[13] T. Borgohain, U. Kumar, and S. Sanyal, "Survey of Security and Privacy Issues of Internet of Things," International Journal of Advanced Networking and Applications, vol. 6, pp. 2372–2378, Feb. 2015.

[14] C. Guimarães, J. Quevedo, R. Ferreira, D. Corujo, and R. L. Aguiar, “Exploring interoperability assessment for Future Internet Architectures roll out,” Journal of Network and Computer Applications, vol. 136, pp. 38–56, 2019, doi: https://doi.org/10.1016/j.jnca.2019.04.008.
[15] G. T. e. E. U. C. Ghali, «"Network-layer trust in named-data networking",» ACM SIGCOMM Computer Communication Review, vol. 44, n° %15, pp. 12 - 19, 2014.
[16] Amar ABANE, “Mise en œuvre des concepts NDN et IoT dans le domaine des Smart Cities : Exemple du parking intelligent,” Mémoire de Fin d’Études de MASTER ACADÉMIQUE, UNIVERSITÉ MOULOUD MAMMERI DE TIZI-OUZOU, TIZI-OUZOU, 2016.
[17] Future Internet Architecture. [Online]. Available: www.nets-fia.net/
[18] Bengt Ahlgren Christian Dannewitz Claudio Imbrenda ,Dirk Kutscher B ̈orje Ohlman (in alphabetical order). A Survey of Information-Centric Networking , February 2, 2011
[19] V. S. D. K. T. J. D. P. M. F. B. N. H. &. B. R. L. Jacobson, «Networking named content,» Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, pp. 1 - 12, 2009.
[20] V. &. S. D. K. Jacobson, « Future Internet architecture: Named data networking,» Proceedings of the 2009 Workshop on Re-Architecting the Internet (ReArch), pp. 1 - 7, 2012.
[21] A. M. I. Z. L. &. Z. B. Afanasyev, «CCNx 1.0 implementation: high-performance named-object networking,» IEEE Communications Magazine, vol. 52, n° %11, pp. 118-124, 2014.
[22] L. Zhang et al. Named Data Networking (NDN) Project. Technical Report NDN-0001, PARC, October 30, 2010.
[23] D. Kim and J. Lee, "An NDN Cache Management for MEC," Appl. Sci., p. 2, 2020
[24] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, “Congestion control in named data networking - A survey,” Computer Communications, vol. 86. pp. 1–11, 2016.
[25] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, “Congestion control in named data networking - A survey,” Computer Communications, vol. 86. pp. 1–11, 2016.
[26] W.So, A.Narayanan, D.Oran, Named Data Networking on a Router: Fast and DoS-resistant forwarding with Hash Tables. 978-1-4799-1640 5/13/$31.00 IEEE 2013.
[27] Faizul Bari, Shihabur Rahman Chowdhury, and Reaz Ahmed, University of Waterloo
Raouf Boutaba, University of Waterloo and Pohang University of Science and Technology
Bertrand Mathieu, Orange Labs « A Survey of Naming and Routing in Information
Centric Networks », Décembre 2012.
[28] A. A. J. B. V. J. k. c. P. C. C. P. L. W. a. B. O. Lixia Zhang, «Named data networking,» ACM SIGCOMM
Computer Communication Review 44, pp. 66-73, 3 July 2014.
[29] A. A. Z. Z. e. L. Z. Y. Yu, «Ndn technical memo: Naming conventions,» Technical report, UCLA, Tech.
Rep, 2014.

[30]C . Yi, A. Afanasyev, L. Wang, B. Zhang, L.Zhang. Adaptive forwarding in Named Data Networking. In ACM SIGCO.[31] Puterman, Martin L. (1994). Markov decision processes: discrete stochastic dynamic programming. Wiley series in probability and mathematical statistics. Applied probability and statistics section. New York: Wiley. ISBN 978-0-471-61977-2.

[32] Schneider, S.; Wagner, D. H. (1957-02-26). "Error detection in redundant systems". Papers presented at the February 26-28, 1957, western joint computer conference: Techniques for reliability on - IRE-AIEE-ACM '57 (Western). New York, NY, USA: Association for Computing Machinery. pp. 115–121. doi:10.1145/1455567.1455587. ISBN 978-1-4503-7861-1.

[33] Bellman, Richard (1958-09-01). "Dynamic programming and stochastic control processes". Information and Control. 1 (3): 228–239. doi:10.1016/S0019-9958(58)80003-0. ISSN 0019-9958.

[34] Sutton, Richard S.; Barto, Andrew G. (2018). Reinforcement learning: an introduction. Adaptive computation and machine learning series (2nd ed.). Cambridge, Massachusetts: The MIT Press. ISBN 978-0-262-03924-6.

[35] R. Sutton and A. Barto, Reinforcement Learning: An Introduction. Cambridge, MA:Weschester Publishing Services, 2018.

[36] H. van Hasselt, A. Guez, and D. Silver, Deep reinforcement learning with double qlearning, in Thirtieth AAAI Conference on Arti cial Intelligence, 2016.

[37] Chong Huang, Student Member, IEEE, Gaojie Chen, Senior Member, IEEE and Yu Gong. Delay Constrained Buffer-Aided Relay Selection in the Internet of Things with Decision-Assisted Reinforcement Learning.

[38] V. François-Lavet, Contributions to deep reinforcement learning and its applications in smartgrids, Ph.D Thesis, Dept. Elect. Eng. Comput. Sci.,Univ. of Liege, Liège, Belgium, 2017.

[39] J. Fan, Z. Wang, Y. Xie, and Z. Yang, A theoretical analysis of deep Q-learning, in Proc. 2nd Conf. Learn. Dyn. Control, vol. 120. PMLR, 2020, pp. 486489.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness,M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland,G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King,D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, Human-level control through deep reinforcement learning, Nature, vol. 518, no. 7540,pp. 529533, Feb. 2015.

[41] Shahid Md. Asif Iqbala,∗, Asaduzzamanb. aDepartment of Computer Science & Engineering, Premier University, 44, Hazari Lane, Kotwali, Chattogram, 4000, Chattogram,

Bangladesh bDepartment of Computer Science & Engineering, Chittagong University of Engineering and Technology, Kaptai Road, Pahartali,Rangunia, 4349, Chattogram, Bangladesh. Cache-MAB: A Reinforcement Learning-based Hybrid Caching Scheme in Named Data

[42] Deep reinforcement learning with experience replay based on sarsa,by D. Zhao, H. Wang, K. Shao, and Y. Zhu, in IEEE Symposium Series on Computational Intelligence, 2016.

[43] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, andK. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in International conference on machine learning, 2016, pp. 19281937.

[44]Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin. Towards biologically plausible deep learning. arXiv preprint arXiv:1502.04156, 2015.

[45]Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In Competition and cooperation in neural nets, pages 267–285. Springer, 1982.

[46]Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1): 1929–1958, 2014.

[47] E. T. d. Silva, J. M. H. de Macedo, and A. L. D. Costa, “NDN content store and caching pol icies: Performance evaluation,” Computers Journal, vol. 11, no. 3, p. 37, 2022. https://doi. org/10.3390/computers11030037”.

[48] L. V. Yovita and N. R. Syambas, “Caching on named data network: A survey and future research,” International Journal of Electrical and Computer Engineering (IJECE), vol. 8, no. 6, pp. 4456–4466, 2018. https://doi.org/10.11591/ijece.v8i6.pp4456-4466”.

[49] E. Aubry, T. Silverston, and I. Chrisment, “Green growth in NDN: Deployment of con tent stores,” in Proceedings of the 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), Rome, Italy, 2016, pp. 1–6.

[50] M. Arlitt, R. Friedrich, and T. Jin, “Performance evaluation of web proxy cache replace ment policies,” in Computer Performance Evaluation (TOOLS 1998), in Lecture Notes in Computer Science, R. Puigjaner, N. N. Savino and B. Serra, Eds., Springer, Berlin, Heidelberg, vol. 1469, 1998, pp. 193–206.

[51] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, “Evaluating content manage ment techniques for web proxy caches,” ACM SIGMETRICS Performance Evaluation Review, vol. 27, no. 4, pp. 3–11, 2000.

[52] Samir Nassane1, Sid Ahmed Mokhtar Mostefaoui, Bendaoud Mebarek, Abdelkader Alm. , “LPCE-Based Replacement Scheme for Enhancing Caching Performance in Named Data Networking iJIM | eISSN: 1865-7923 | Vol. 18 No. 16 (2024).

[53] N. Laoutaris, H. Che, and I. Stavrakakis, “The LCD interconnection of LRU caches and its analysis,” Performance Evaluation, vol. 63, no. 7, pp. 609–634, 2006, doi: .

[54] N. R. Syambas, H. Situmorang, and M. A. P. Putra, “Least recently frequently used replacement policy in named data network,” in 2019 IEEE 5th International Conference on Wireless and Telematics (ICWT), Yogyakarta, Indonesia, 2019, pp. 1–4.

[55] G. Karakostas and D. N. Serpanos, “Exploitation of different types of localities for Web caches,” in Proceedings ISCC 2002 Seventh International Symposium on Computers and Communications, Taormina-Giardini Naxos, Italy, 2002, pp. 207–212.

[56] P. Singh, R. Kumar, S. Kannaujia, and N. Sarma, “Adaptive replacement cache policy in named data networking,” in 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India, 2021, pp. 1–5.

[57]P. Singh, R. Kumar, S. Kannaujia, and N. Sarma, “Adaptive replacement cache policy in named data networking,” in 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India, 2021, pp. 1–5.

[58] Reward-based learning, model-based and model-free QJM Huys, A Cruickshank, P Seriès Encyclopedia of Computational Neuroscience, 2014.

[59] Janith K. Dassanayake, Minxiao Wang, Muhammad Z. Hameed, Ning Yang. Multi-Agent Deep-Q Network-Based Cache Replacement Policy for Content Delivery Networks.

[60] Dueling network architectures for deep reinforcement learning ,Z Wang, T Schaul, M Hessel… - International …, 2016.

[61] Mohit sewak. Deep Q Network (DQN), Double DQN, and Dueling DQN: A Step Towards General Artificial Intelligence, 2019 https://www.researchgate.net/publication/334070121.

[62] Meriem Bahi, Mohamed Batouche. Deep Learning for Ligand-Based Virtual Screening in Drug Discovery.oct 2019.

[63] Ankur Pandey · Praveen Kumar Mannepalli · Manish Gupta · Ramraj Dangi · Gaurav Choudhary5. A Deep Learning-Based Hybrid CNN-LSTM Model for Location-Aware Web Service Recommendation.2024.

https://doi.org/10.1007/s11063-024-11687-w

[64] Steffen G. Scholz, Ahmed Elkaseer, Saeed Mohsen Industry .4.0-Oriented Deep Learning Models for Human Activity Recognition.2021. https://www.researchgate.net/publication/356018554.

[65]D. Karmiani, R. Kazi, A. Nambisan, A. Shah, and V. Kamble, ‘‘Comparison of predictive algorithms: Backpropagation, SVM, LSTM and Kalman filter for stock market,’’ in Proc. Amity Int. Conf. Artif. Intell. (AICAI), Feb. 2019, pp. 228–234.

[66]Dangi R, Lalwani P, Mishra MK (2023) 5g network traffic control: a temporal analysis and forecasting of cumulative network activity using machine learning and deep learning technologies. Int J Ad Hoc Ubiquitous Comput 42(1):59–71.

[67] Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, AlAmidie M, Farhan L (2021) Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. J big Data 8:1–74.

[68] Deepali D. Ahir, Sagar B. Shinde . Caching Simulators for Content Centric Networking. Pune University, M .E. S College of Engineering Wadia Campus, 19, Bund Garden, V.K. Joag Road, Pune, India.

[69] Lada A. Adamic1 Bernardo A. Huberman “Zipf’s law and the Internet”, Glottometrics 3, 2002,143-150.

[70] Fagan, Stephen; Gençay, Ramazan (2010). "An introduction to textual econometrics". In Ullah, Aman; Giles, David E.A. (eds.). Handbook of Empirical Economics and Finance. CRC Press. pp. 133–153, esp.&nbps, 139. ISBN 9781420070361. For example, in the Brown Corpus, consisting of over one million words, half of the word volume consists of repeated uses of only 135 words.

[71] Michele Tortelli, Luigi Alfredo Grieco and Gennaro Boggia DEI, Politecnico di Bari (Italy) “Performance Assesment of Routing Strategies in Named Data Networking” GTTI 2013 Session on Telecommunication Networks.

[72] Ahlgren,B.,Dannewitz,C.,Imbrenda,C.,&Kutscher,D.(2012).A survey of information-centric networking. IEEE Communications Magazine, 50(7), 26–36.

[73] Brito, G. M., Velloso, P. B., & Moraes, I. M. (2013). Information centric networks: A new paradigm for the internet (1st ed.). John Wiley\& Sons.

[74] Chen, X., Zhang, G., & Cui, H. (2018). Investigating route cache in named data networking. IEEE Communications Letters, 22(2), 296–299.