informatics

MDPI

Article

A Survey of Deep Learning for Data Caching in Edge Network

Yantong Wang * and Vasilis Friderikos

Center for Telecommunications Research, Department of Engineering, King's College London, London WC2R 2LS, UK; vasilis.friderikos@kcl.ac.uk

* Correspondence: yantong.wang@kcl.ac.uk

Received: 21 August 2020; Accepted: 2 October 2020; Published: 13 October 2020

Abstract: The concept of edge caching provision in emerging 5G and beyond mobile networks is a promising method to deal both with the traffic congestion problem in the core network, as well as reducing latency to access popular content. In that respect, end user demand for popular content can be satisfied by proactively caching it at the network edge, i.e., at close proximity to the users. In addition to model-based caching schemes, learning-based edge caching optimizations have recently attracted significant attention, and the aim hereafter is to capture these recent advances for both model-based and data-driven techniques in the area of proactive caching. This paper summarizes the utilization of deep learning for data caching in edge network. We first outline the typical research topics in content caching and formulate a taxonomy based on network hierarchical structure. Then, many key types of deep learning algorithms are presented, ranging from supervised learning to unsupervised learning, as well as reinforcement learning. Furthermore, a comparison of state-of-the-art literature is provided from the aspects of caching topics and deep learning methods. Finally, we discuss research challenges and future directions of applying deep learning for caching.

Keywords: deep learning; content caching; network optimization; edge network

1. Introduction

Undoubtedly, future 5G and beyond mobile communication networks will have to address stringent requirements of delivering popular content at ultra high speeds and low latency due to the proliferation of advanced mobile devices and data rich applications. In that ecosystem, edge-caching has received significant research attention over the last decade as an efficient technique to reduce delivery latency and network congestion especially during peak-traffic times or during unexpected network congestion episodes by bringing popular data closer to the end users. One of the main reasons for enabling edge caching in the network is to reduce the number of requests that traverse the access and core mobile network, as well as reducing the load at the origin servers that would have to, otherwise, respond to all requests directly in absence of edge caching. In that case, popular content and objects can be stored and served from edge locations, which are closer to the end users. This operation is also beneficial from the end user perspective since edge caching can dramatically reduce the overall latency to access the content and increase in the sense of overall user experience. It is also important to note that the notion of popular content means that the requests of top 10% of video content on the Internet account for almost 80% of all traffic, which relates to multiple requests from different end users of the same content [1].

Recently, deep learning (DL) has attracted significant attention from both academia and industry and has been applied to diverse domains, like self-driving, medical diagnosis, and playing complex games, such as Go [2]. DL has also made their way into communication areas [3]. In this paper, we pay attention to the application of DL in caching policy. Though there are some earlier surveys

related to machine learning applications, they either focus on general machine learning techniques for caching [4–6], or concentrate on overall wireless applications [7–9]. The work [3] provides a big picture of applying machine learning in wireless communications. In Reference [4], the authors consider the machine learning on both caching and routing strategy. A comprehensive survey on machine learning applications for caching content in edge networks is provided in Reference [5]. The researchers in Reference [6] provide a survey about machine learning on mobile edge caching and communication resources. On the other hand, Reference [7] overviews how artificial neural networks can be employed for various wireless network problems. The authors in Reference [8] detail a survey on deep reinforcement learning (DRL) for issues in communications and networking. Reference [9] presents a comprehensive on deep learning applications and edge computing paradigm. Our work can be distinguished from the aforementioned papers based on the fact that we focus on the deep learning techniques on content caching and both wired and wireless caching are taken into account. Our main contributions are listed as follows:

- We classify the content caching problem into Layer 1 Caching and Layer 2 Caching. Each layer caching consists of four tightly coupled subproblems: where to cache, what to cache, cache dimensioning, and content delivery. Related researches are provided accordingly.
- We present the fundamentals of DL techniques which are widely used in content caching, such as convolutional neural network, recurrent neural network, actor-critic model-based deep reinforcement learning, etc.
- We analyze a broad range of state-of-the-art literature which use DL to content caching. These papers are compared based on the DL structure, layer caching coupled subproblems, and the objective of DL in each scenario. Then, we discuss research challenges and potential directions for the utilization of DL in caching.

The rest of this survey is organized as follows (as illustrated in Figure 1). Section 2 presents the categories of content caching problem. Section 3 reviews typical deep neural network structures. In Section 4, we list state-of-the-art DL-based caching strategies and their comparison. Section 5 debates challenges, as well as potential research directions. In the end, Section 6 concludes this paper. For better readability, the abbreviations in this paper is listed as Table 1 shows.

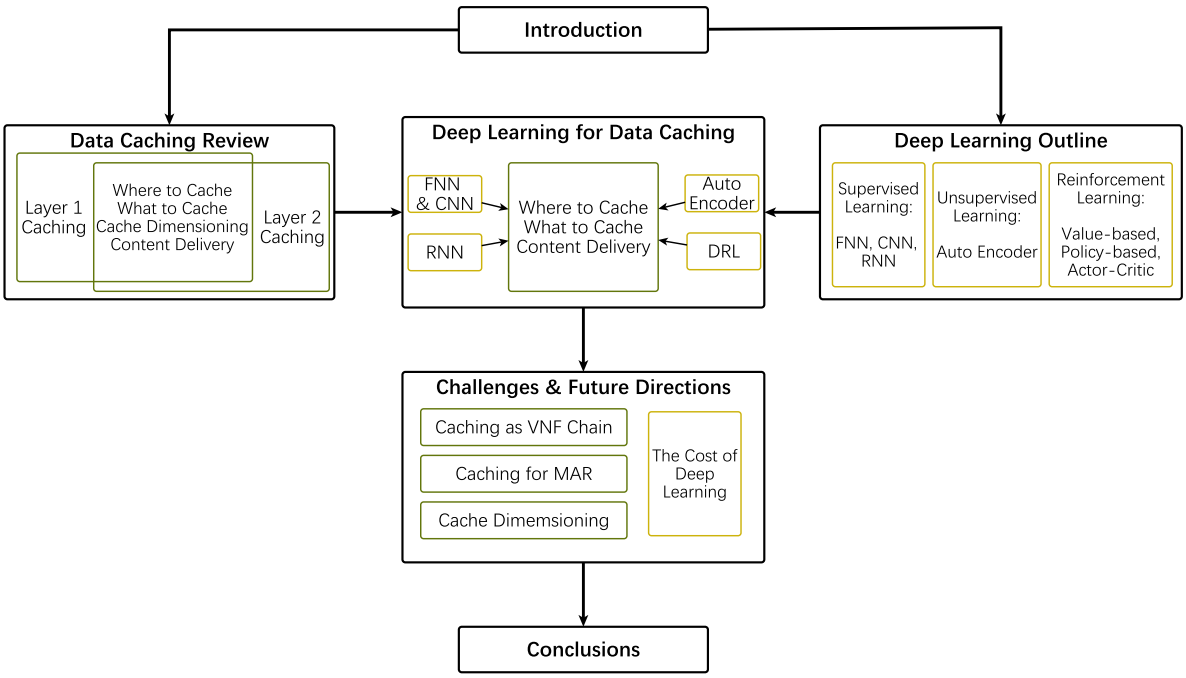

Figure 1. Survey architecture.

Table 1. List of abbreviations.

Abbr.	Description	Abbr.	Description
3C	Computing, Caching and Communication	A3C	Asynchronous Advantage Actor-Critic
BBU	Baseband Unit	CCN	Content-Centric Network
CNN	Convolutional Neural Network	CoMP-JT	Coordinated Multi Point Joint Transmission
CR	Content Router	C-RAN	Cloud-Radio Access Network
CSI	Channel State Information	D2D	Device to Device
DDPG	Deep Deterministic Policy Gradient	DL	Deep Learning
DNN	Deep Neural Network	DQN	Deep Q Network
DRL	Deep Reinforcement Learning	DT	Digital Twin
ED	End Device	ES	Edge Server
ETSI	European Telecommunication Standardization Institute		
ESN	Echo-State Network	FIFO	First In First Out
FNN	Fully-Connected Neural Network	FBS	Femto Base Station
GRU	Gated Recurrent Unit	ICN	Information-Centric Network
IoT	Internet of Things	LFU	Least Frequently Used
LP	Linear Programming	LRU	Least Recently, Used
LSTM	Long Short-Term Memory	MAR	Mobile Augmented Reality
MD	Mobile Device	MILP	Mixed Integer Linear Programming
MBS	Macro Base Station	NFV	Network Function Virtualization
PNF	Physical Network Function	PPO	Proximal Policy Optimization
QoE	Quality of Experience	RL	Reinforcement Learning
RNN	Recurrent Neural Network	RRH	Remote Radio Head
SAE	Sparse Auto Encoder	SDN	Software Defined Network
Seq2Seq	Sequence to Sequence	SNM	Shot Noise Model
TRPO	Trust Region Policy Optimization	TTL	Time to Live
VNF	Virtual Network Function	WSN	Wireless Sensor Network

2. Data Caching Review

The paradigm of data caching in edge networks is illustrated in Figure 2. Similar to [10], the scope of edge in this paper is along the path between end user and data server, which contains Content Router (CR), Macro Base Station (MBS), Femto Base Station (FBS), and End Device (ED). In the context of Cloud-Radio Access Network (C-RAN) [11], both baseband unit (BBU) and remote radio head (RRH) are considered as potential caching candidates to hosting content, where the BBUs are clustered as a BBU pool centrally and RRHs are deployed near BS's antenna distributively. According to the hierarchical structure of edge network, the data caching is classified into two categories: Layer 1 Caching and Layer 2 Caching. In this section, we illustrate the typical research topics in these two areas.

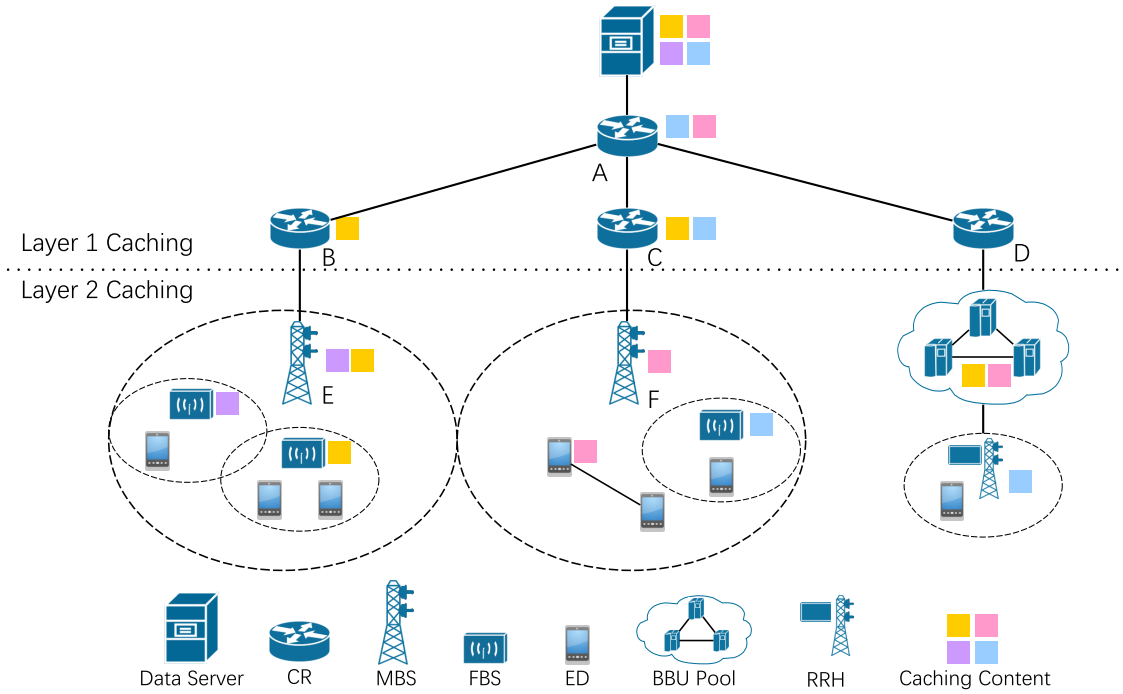

Figure 2. Data caching in edge network.

2.1. Layer 1 Caching

In Layer 1 Caching, the popular content is considered to be hosted in CRs. In the context of Information-Centric Network (ICN), CR plays dual roles both as a typical router (i.e., data flow forwarding) and content store (i.e., local area data caching facility). Generally, the CR is connected via wired networks. Layer 1 Caching consists of four tightly coupled problems: where to cache, what to cache, cache dimensioning, and content delivery [12].

Where to cache focuses on selecting the proper CRs to host the content. For instance, in Figure 2, contents replicas can be placed in lower hierarchical level CRs, such as router B and C, as the mean of reducing transmission cost but with extra cost pay for hosting contents; reversely, consolidating caching in CR A can be adopted to saving caching cost at the expense of more transmission cost and has the risk of expiring end users' delay requirement. Here, the caching/hosting cost is the cost to deploy the content, which could be measured by space utilization, energy consumption, or other metrics. The transmission cost represents the price for delivering the content from cached CR (or data server) to end user and is estimated via the number of hops generally. Where to cache problem usually has been modeled as a Mixed Integer Linear Programming (MILP):

$$\min_{x} \; c^T x, \tag{1a}$$

$$\text{s.t.} \quad Ax \leq b, \tag{1b}$$

$$x \in \{0, 1\}, \tag{1c}$$

$$\text{or} \quad x \geq 0, \tag{1d}$$

where x is the decision variable. Normally it is a binary variable indicating the CR assignment. In special cases, with the aim of modeling or linearization, some non-binary auxiliary variables are introduced as constraint (1d) shows. If taking caching a part of a file not the complete into consideration, the decision variable x is a continuous variable representing the segments which are hosted in the CR, then constraint (1c) becomes $x \in [0, 1]$ and the MILP model turns to linear programming (LP). There are many papers allocating contents via MILP with different objectives and limitations. The authors in Reference [13] propose a model to minimize the user delay and load balancing level of CRs with the

satisfaction of cache space. The work in Reference [14] considers a trade-off between caching and transmission cost with cache space, link bandwidth, and user latency constraints. In Reference [15], an energy efficient optimization model is constructed consisting of caching energy and transport energy. Reference [16] provides more details of the mathematical model and related heuristic algorithms in caching deployment of wired networks.

What to cache concentrates on selecting the proper contents in CRs for the purpose of maximizing the cache hit ratio. Via exploiting the statistical patterns of user requests, the popularity of requested information and user preference can be forecasted and play a very significant role in determining caching content. On the one hand, from the view of aggregated request contents, researchers propose many different models and algorithms for popularity estimation. One widely used model in web caching is the Zipf model based on the assumption that the content popularity is static and each users' request is independent [17]. However, this method fails to reflect the temporal and spatial correlations of the content, where the temporal correlation reflects the popularity varies over time and the spatial correlation represents the content preference is different on the geographical area and social cultural media. A temporal model named the shot noise model (SNM) is built in Reference [18] which enables users to estimate the content popularity dynamically. Inspired by SNM, the work in Reference [19] considers both spatial and temporal characteristics during caching decisions. On the other hand, from the view of a specific end user during a certain period, caching his/her preference content (may not be popular in the network) can also help to reduce the traffic flow. Many approaches in recommendation systems can be applied in this case [20]. Another aspect of what to cache problem is the designing of cache eviction strategies when storage space faces the risk of overflow. Depending on the life of caching contents, these policies can be divided into two categories roughly: one includes first in first out (FIFO), least frequently used (LFU), least recently used (LRU), and randomized replacement, where the contents would not be removed until no more memory is available; the other one is called time to live (TTL) strategy, where the eviction happens once the related timer expires. Reference [21] presents an analytic model for hit ratio in TTL-based cache requested by independent and identically distributed flows. It worth noting that in Reference [21], the TTL-based cache policy is used for the consistency of dynamic contents instead of contents replacement. In Reference [22], the authors introduce a TTL model for cache eviction and the timer is reset once related content cache hit happens.

Cache dimensioning highlights how much storage space to be allocated. Benefitting from the softwarization and virtualization technologies, the cache size in each CR or edge cloud can be managed in a more flexible and dynamical way, which makes the cache dimensioning decisions become an important feature in data caching. Technically, the cache hit ratio rises with the increase of cache memory and consequently eases the traffic congestion in the core network. However, excessive space allocation would waste the resource, like energy, to support the caching function. Hence there is a trade-off between cache size cost and network congestion. Economically, taking such a scenario into consideration: a small content provider wants to rent service from a Content Delivery Network (CDN) provider, such as Akamai or Huawei Cloud, and there is also a balance between investment saving and network performance. In Reference [23], the proper cache size of individual CR in Content-Centric Network (CCN) is investigated via exploiting the network topology. In Reference [24], the authors consider the effect of network traffic distribution and user behaviors when designing cache size.

Content delivery considers how to transform the caching content to the requested user. The delivery traffic embraces single cache file downloading and video content streaming and the metrics for these two scenarios vary. Regarding file downloading, the content cannot be consumed until the delivery is completed. Therefore, the downloading time of the entire file is viewed as a metric to reflect the quality of experience (QoE). For video streaming, especially for those large video split into several chunks, the delay limitation only works on the first chunk. In that case, delivering the first chunk in time and keep the smooth transmission of the rest chunks are the key aims [25]. Apart from those measuring metrics, another problem in content delivery is the routing policy. In CCN [26], one implementation of ICN architecture employs a flooding-based name routing protocol to publish

the request among cached CRs. On one hand, flooding strategy simplifies the designing complexity and reduce the maintaining cost particularly in an unstable scenario; on the other hand, it costly wastes bandwidth resources. In Reference [27], the authors discuss the optimal radius in scoped flooding. The deliver route is often considered jointly with where to cache problem, in which the objective function (1a) includes both deployment and routing cost.

2.2. Layer 2 Caching

Contrast to Layer 1 caching in wired connection, Layer 2 caching considers implementing caching techniques in the wireless network. Though both of them need to solve where to cache, what to cache, cache dimensioning, and content delivery problems, wireless caching is more challenging, and some mature strategies in wired caching cannot be migrated directly to the wireless case. Some reasons come from the listed aspects: the resources in the wireless environment, such as caching storage and spectrum, are limited compared with CRs in Layer 1 Caching; the mobility of end users and dynamic network topologies are also required to be considered during the design of caching strategies; moreover, the wireless channels are uncertain since they can be affected by fading and interference.

In wireless caching, where to cache focus on finding the proper candidates among MBS, FBS, ED, even BBU pool and RRU in C-RAN to host the content. Caching at MBS and FBS can alleviate backhaul congestion since end users obtain the requested content from BS directly instead of from CR via backhaul links. Compared with FBS, MBS has a wider coverage and typically, there is no overlap among different MBSs [28]. As mentioned above, the caching space in BSs is limited and it is impractical to cache all popular content. With the aim of improving the cache-hit ratio, a MILP-modeled collaborative caching strategy among MBSs is proposed in Reference [29]. If the accessed MBS does not host the content, the request will be served by a neighbor MBS which caches the file rather than by the data server. For FBS caching, a distributed caching method is presented in Reference [30] and the main idea is that the ED locating in the FBS coverage overlap is able to obtain contents from multiple hosters. Caching at ED can not only ease backhaul congestion but also improve the area spectral efficiency [28]. When the end user requests, he/she would be severed by the local storage if the content is precached in his/her ED or by adjacent ED via Device to Device (D2D) communication if the content is host accordingly. In Reference [31], the authors model the cache-enabled D2D network as a Poisson cluster process, where end users are grouped into several clusters and the collective performance is improved. Individually, caching the interested contents for other users affects personal benefit. In Reference [32], a Stackelberg game model is applied to formulate the conflict among end users and a related incentive mechanism is designed to encourage content sharing. For the case of cache-enabled C-RAN, caching at BBU can ease the traffic congestion in the backhaul, while caching at RRH can reduce the fronthaul communication cost. On the other hand, caching all at BBU raises the signaling overhead of the BBU pool, while at RRH weakens the processing capability. Therefore, where to cache the content in C-RAN makes a substantial contribution to balancing the signal processing capability at the BBU pool and the backhaul/fronthaul costs [28]. The work in Reference [33] investigates caching at RRHs with jointly considering cell outage probability and fronthaul utilization. Due to the end users' mobility, the prediction/awareness of user moving behavior also influences the proper hoster selection. There are some researches exploiting user mobility in cache strategy designing, like Reference [34,35].

Similar to Layer 1, what to cache decision, as well as eviction policy, of Layer 2 depends on the accurate prediction on content popularity or user preference in the proactive caching method. The content popularity contains the feature of temporal and spatial correlations, which has already been described in Layer 1 Caching. In Layer 2 caching, the proper spatial granularity in popular contents estimation needs to take special attention [36]. For example, the coverage of MBS and FBS are different, which makes the popularity in MBS and FBS are different, as well. The former is based on a large number of users' behaviors, but the individual may prefer specific content categories. For small cells, the preference estimation requires more accurate information, like historical data [28]. In order to

capture the temporal and spatial dynamics of user preference, many different deep learning based algorithms are proposed, which will be illustrated in Section 4.

Cache dimensioning in Layer 2 Caching has more complicated factors need to be considered, not only including the network topology and content popularity as Layer 1 Caching, but also containing backhaul transmission status and wireless channel features. The proper cache size assignment is studied in the scenario of backhaul limited cellular network [37]. It also provides the closed-form boundary of minimum cache size in one cell case. In the case of dense wireless network, the work in Reference [38] quantifies the minimum required cache to achieve the linear capacity scaling of network throughput. The authors of Reference [39] also consider the scenario of dense networks. They derive the closed-form of the optimal memory size which can reduce the consumption of backhaul capacity, as well as guarantee wireless QoS.

According to the number of transmitters and receivers, we divide the content delivery in Layer 2 caching into three categories: one candidate serves one end user, such as unicast and D2D transmission; one candidate serves multiple users, like multicast; coordinated delivery including multiple transmitters serve one or more receivers, like coordinated multi-point joint transmission (CoMP-JT). Once the requested content is cached locally, BS can serve the end user via unicast or the adjacent device shares the contents by implementing D2D transmission. The concurrent transmission has the risk of co-channel interference in densely deployed networks. In the D2D network, link scheduling is introduced to select subsets of links to transmit simultaneously [28]. In order to improve spectral efficiency, multicast is applied in content delivery when serving multiple requests simultaneously with the same content. Therefore, there is a trade-off between spectral efficiency and service delay. For the aim of serving more users in one transmission, as well as higher spectral efficiency, the BS will wait to collect enough requirement for the same content, which makes the first request a long waiting time. An optimal dynamic multicast scheduling is proposed in Reference [40] to balance these two factors. Multicast can also serve multiple requests with different contents. In Reference [41], the authors provide a coded caching scheme which requires the communication link is error-free and each user caches a part of its own content and partial for other users. Then, BS multicasts the coded data to all users. Each user can decode his own requested content by XOR operation between the received data and the precached other users' file. However, the coding complexity increases exponentially as the number of end users grows. The CoMP-JT can improve the spectral efficiency, as well via sharing channel state information (CSI) and contents among BSs, but it also needs high-capacity backhaul consumption for exchanging data. In C-RAN, the BBUs are centralized in the BBU pool, which makes communication among BSs very efficiency. Reference [42] designs CoMP-JT in C-RAN for the purpose of minimizing power consumption with limitations of transmission energy, link capacity and requested QoS.

3. Deep Learning Outline

As Figure 3 shows, some typical deep neural network (DNN) methods are stated. These models are classified into three categories depending on the training methods: supervised learning, unsupervised learning, and reinforcement learning.

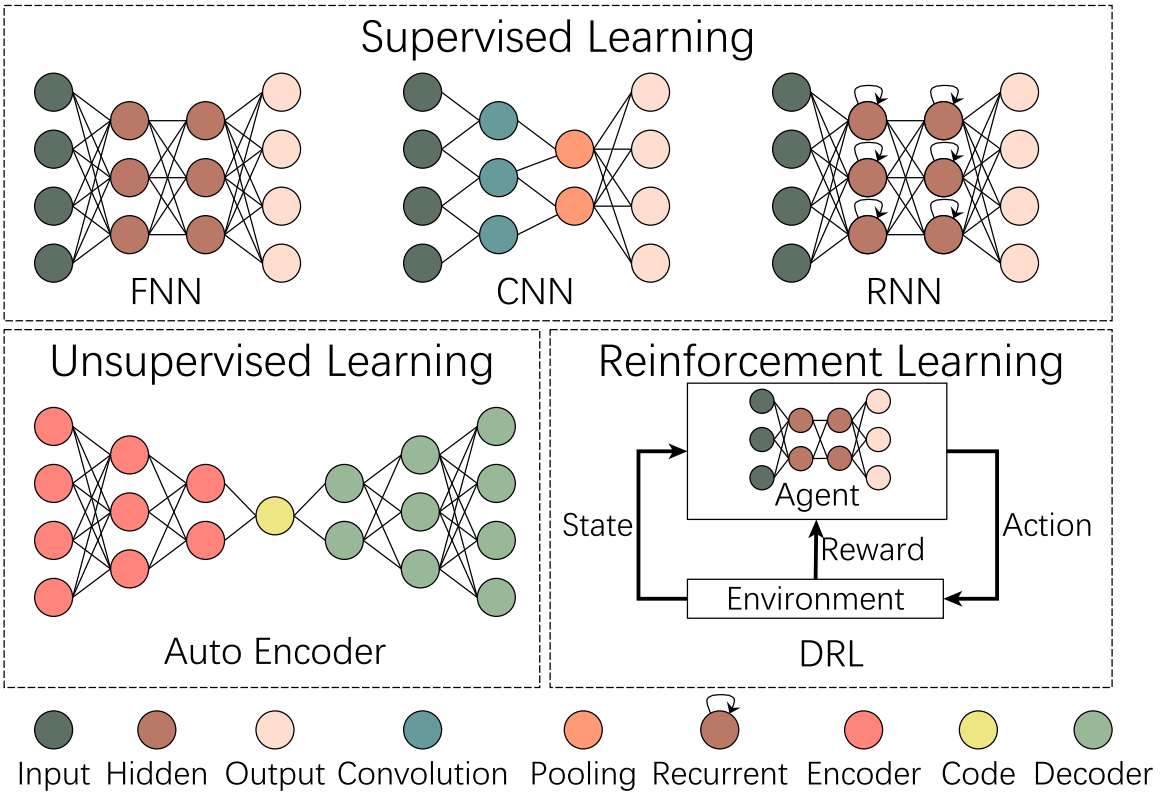

Figure 3. Typical deep neural network (DNN) structures.

3.1. Fully-Connected Neural Network (FNN)

FNN is a kind of feedforward neural network whose information propagation direction is forward and there is no cycle in neurons. As the name indicates, the connection between two adjacent layers of FNN is filled. The hidden layer is applied to extract features of the input vector, and then feed the output layer, which works as a classifier. Since each neuron of FNN is fully connected, each digit of input vector makes contributions to the final output. That is to say, FNN hosts the ability to extract global features. On the other hand, it lacks local pattern attention without data preprocessing, such as augmentation. According to the Universal Approximation Theorem, FNN has the ability to approximate any closed and bounded function with enough neurons in hidden layer [43]. Though FNN is very powerful, it gets into trouble when dealing with real-world tasks, such as image recognition, due to enormous number of weight parameters (because of being fully connected).

3.2. Convolutional Neural Network (CNN)

As aforementioned, the enormous number of weight parameters pose a significant challenge in the training complexity of FNNs. In order to overcome this drawback, whilst being inspired by the idea of receptive field in biology, CNN is proposed with the structure of stacked convolutional layer and pooling layer, followed by some fully connected layers. The convolutional layer employs sliding convolutional filters to the input vector, and the filter's weights do not change during processing, a.k.a. weight sharing. The pooling layer applies downsampling operation, usually through the maximum or mean pooling. In this respect, both weight sharing and pooling reduce the number of training parameters. It is worth noting that the downsampling operation also takes the risk of dropping some information, which lowers the association between the local and global patterns, especially for the case of info-intensive task (i.e., each element of a matrix or each pixel of an image plays an important role in final object detection). For example, there is no pooling layer in Alpha-Go's neural architecture [44]. Generally, CNN tends to contain deeper layers and smaller convolutional filters, and the structure is likely to be a fully convolutional network [45], reducing the ratio of

pooling layers, as well as fully connected layers. Taxonomically, CNN belongs to feedforward neural network and is the mainstream deep model in computer vision. The reasons come from that in image recognition: (1) the local pattern makes more contribution in classifications; (2) the same pattern appears in different positions; and (3) downsampling some pixels will not affect the classification. The convolutional layer fits reason (1) and (2), while the pooling layer suits reason (3). CNN is also employed in natural language processing. Including CNN and FNN, one of the limitations of the feedforward neural network is that the output only depends on current input vectors. So, it is hard to deal with sequential tasks.

3.3. Recurrent Neural Network (RNN)

In order to deal with sequential tasks and using historical information, RNN employs neurons with self feedback in hidden layers. Unlike the hidden neuron in feedforward neural network, the output of recurrent neuron depends on both the current output of precious layer and last hidden state. Compared with FNN approximates any continues functions, RNN with Sigmoid activation function can simulate a universal Turing Machine and has the ability to solve all computational problems [46]. Theoretically, RNN can deal with any long time series. Practically, RNN has the risk to suffer from long-term dependencies problem [43] including gradient exploding and vanishing. Additionally, RNN has more parameters waiting to be trained due to adding recurrent weights. In the following, we introduce some RNN variants as Figure 4 shows.

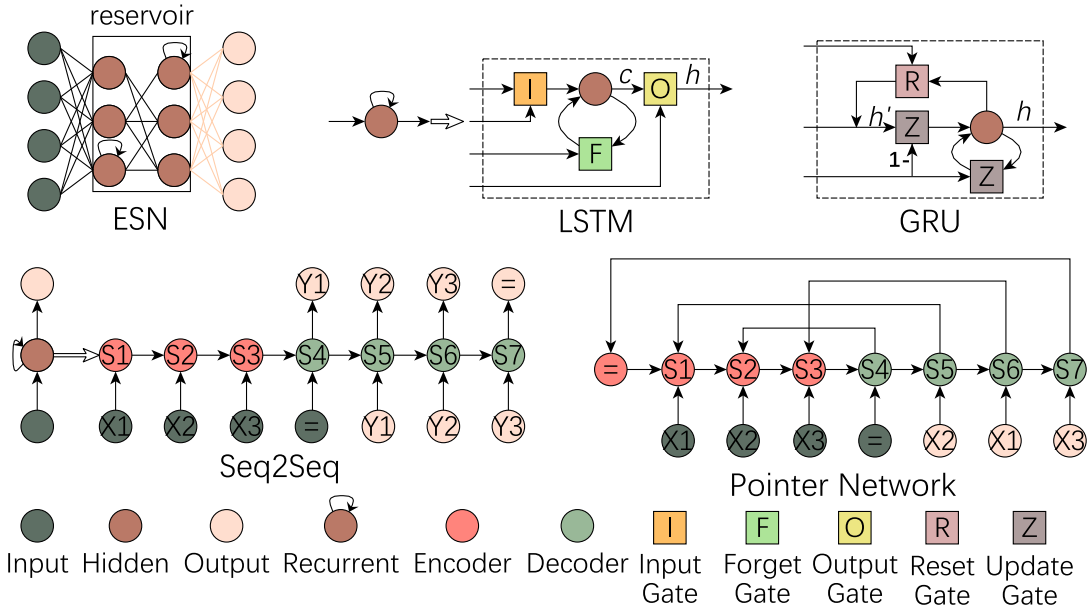

Figure 4. Recurrent Neural Network (RNN) variants.

3.3.1. Echo-State Network (ESN)

As aforementioned, simple RNN contains more parameters in the training step, where the recurrent weights and input weights are difficult to learn [43]. The basic idea of ESN is fixing these two kinds of weights and only learn the output weights (as links highlighted in Figure 4). The hidden layer is renamed as reservoir in ESN, where the neurons are sparsely connected and the weights are randomly assigned. The recurrent weights keep constant so the information of previous moments is stored in the reservoir with constant weight, like voice echoing. To avoid echo explosion, the eigenvalues of recurrent weights should not be greater than 1. On the one hand, the preassigned weights of input and reservoir layers reduce the number of training parameters then accelerate the training process; on the other hand, since the connectivity and weight value are random, so the result is unlikely to be optimal.

ESN has been successfully utilized in speech processing, stock price prediction, language modeling, etc. However, when it comes to high dimensional temporal task, ESN is less competitive because only the output weights are flexible during training. In order to extract different features, the reservoir should be enough large, which challenges the ESN design.

3.3.2. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)

Recently, an efficient way to cope with long-term dependencies in practical is employing gated RNN, including LSTM [43]. Then, we compare with the recurrent neuron in simple RNN: Internally LSTM introduces three gates to control signal propagation, where input gate I decides the partition of input signal to be stored, forget gate F controls ratio of last moment memory to be kept until next period (the name "forget gate" may be a little misleading because it actually represents the ratio to be remembered), and output gate O influences the proportion of current state to be delivered; Externally LSTM has four inputs embracing one input signal and three control signals for three gates. All these four signals are derived via the calculation of current network input and last moment delivered state (i.e., the output of last time state, called hidden state h_{t-1}). The data flow before the output gate is named inner state c_t. Intuitively, c_t contains something new learned via the input and the previous knowledge through h_{t-1}. In other words, some old memories still have an influence on the current decision so the gradient vanishing is eased. The side effect of LSTM is that the implementation of gate mechanism is very complicated, which makes both training and testing slower. Because of the temporal causal, LSTM or common RNN does not support parallel training like CNN. One explanation of the name "Long Short-Term Memory" is given by Xipeng Qiu in the book "Neural Networks and Deep Learning" (https://nndl.github.io/): the hidden state h stores the historical information, which is viewed as a kind of memory. In the simple RNN, this state is updated at each time slot, which is called short-term memory. Besides, the parameter of neural network is updated at each iteration, which is regarded as long-term memory. The life of h is longer than the short-term memory and that is why it is named LSTM.

A simpler version is employing gated recurrent unit (GRU) [47]. On one hand, the role of input gate and forget gate in LSTM are complementary. So, GRU combines these two gates and utilizes a single gate Z to control the balance between the previous memory and newly learned knowledge. On the other hand, GRU removes the output gate since both hidden state h and inner state c of LSTM contain the information of last state. Moreover, the candidate state h' in GRU depends on current input and the result of reset gate R. Compared with LSTM, GRU has fewer parameters, which indicates its training is faster than LSTM.

3.3.3. Seq2Seq and Pointer Network

A typical application of RNN is converting one sequence to another sequence (Seq2Seq, also called encoder-decoder model), such as machine translation. As shown in Figure 4, Seq2Seq, the RNN is extended through the timeline, where the input series is $X = \{X1, X2, X3\}$ and the output is $Y = \{Y1, Y2, Y3\}$. The equal symbol "=" represents the end of a sequence. Conventionally, the output of Seq2Seq architecture is a probability distribution of output dictionary. However, it cannot deal with the kind of problems that the size of output relies on the length of input due to the fixed output dictionary. In Reference [48], the authors modify the output to be the distribution of input sequence, which is analogous to pointers in C/C++. Pointer network has been widely used in text condensation. Strictly speaking, the ESN and LSTM are the variants of RNN from the view of network constructions, while Seq2Seq and pointer network are from application modes. Moreover, in some specific cases, like require running time guarantee, the hidden layers of Seq2Seq can be implemented by CNN instead of RNN [49].

3.4. Auto Encoder

Auto Encoder is a stack of two neural networks (NNs), named encoder and decoder, respectively, where the former tries to learn the representative characteristics of input and generate a related code, and the later reads the code and reconstructs the original input. In order to avoid the auto encoder simply copying the input, some restrictions are considered, like the dimension of code is smaller than input vector [43]. The quality of auto encoder can be measured via reconstruction error, which estimates the similarity between input and output. In most cases, the auto encoder is used for the proper representation of the input vector so the decoder part is removed after unsupervised training. The code can be employed as input for further deep learning models. Additionally, the code after dimensional reduction is easier for data visualizing analysis compared with the original high dimensional input data. Because it is unsupervised learning, the auto encoder is expensive to pretrain with big data.

3.5. Deep Reinforcement Learning (DRL)

Reinforcement Learning (RL) is a Markov Decision Process represented by a quintuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ is the state space controlled by environment; $\mathcal{A}$ is the action space determined by agent; $\mathcal{P}$ is the state transition function measuring the probability of moving to a new state s_{t+1} given previous state s_t and action a_t; R is reward function calculated by environment considering state and action; γ is a discount factor for estimating total reward. During the interaction between agent and environment, agent observes current state s_t from environment and then takes action a_t following its policy π. The environment moves to a new state s_{t+1} stochastically based on $\mathcal{P}(s_t, a_t)$ and returns a reward r_t to agent. The RL's aim is finding the policy π to maximum accumulated reward $\sum_t \gamma^t r_t$. In the early stage, RL focuses on scenarios whose $\mathcal{S}$ and $\mathcal{A}$ are discrete and limited. So, the agent can use a table to record this information. Recently, some tasks have enormous discrete states and actions, such as playing go and even continuous value, such as self-driving, which makes table recording impractical. In order to solve this, DRL combines RL and DL, where RL defines the problem and optimization object; DL models the policy and the reward expectation. Depending on the roles of DNN in DRL, we classify the DRL into 3 categories as follows.

3.5.1. DNN as Critic (Value-Based)

In value-based method, DNN does not get involved with policy decision but estimates the policy performance. Two functions are introduced for the measurement: $V^\pi(s)$ represents the reward expectation of policy π starting from state s; $Q^\pi(s, a)$ illustrates the reward expectation of policy π starting from state s and taking action a. In addition, $V^\pi(s)$ is the expected value of $Q^\pi(s, a)$. If we can estimate $Q^\pi(s, a)$, the policy π can also be improved by choosing the action a^* hold $Q^\pi(s, a^*) \geq V^\pi(s)$. As aforementioned, the conventional Q Learning scheme is impractical when dealing with tasks containing enormous discrete states and actions. So, the DNN employed in agent is approximating function $Q^\pi(s, a)$, where the inputs are state s and action a and output is the estimated value $Q^\pi(s, a)$. There are some representative critic methods, like Deep Q Networks (DQN) [50,51] and its variants Double DQN [52], Dueling DQN [53], etc.

As shown in the left side of Figure 5, the DNN Q is applied to estimate the state-action reward $Q^\pi(s, a)$ in the DQN scheme. There are two main problems in training step if we only employ a single DNN: (1) the target is unstable, where the objective function/loss function for training DNN parameters depends on these parameters themselves; (2) these training samples are strongly correlated instead of independent (for example, samples of playing video games are generated from continuous frames), which makes the gradient descent towards a deterministic direction then the training process has a high risk to be not convergent. To deal with the aforementioned problems, DQN takes two measures accordingly: (1) Freezing target DNN Q', in Figure 5, we put a lock symbol, indicating the parameters are fixed during several training steps, to keep the learning objective steady;

(2) Experience replay memory D, which consists of interactions between the client and the environment. Initially, the target DNN Q' is a replica of DNN Q. In each training episode, the client observes a current state from the environment, executes an action depending on the estimation of Q, and receives a reward. Those interactions (i.e., the current state s, action a, reward r, and state after transition ss) are stored in D. Then, target DNN Q' samples random batches of interactions from D and calculates the reward value q' as follows:

$$q' = \begin{cases} r, & \text{if } ss \text{ is a terminated state} \\ r + \gamma \cdot \max_{a'} Q'(ss, a'), & \text{otherwise} \end{cases}. \quad (2)$$

The loss function is defined as the mean square error: $(q' - Q(s,a))^2$. After several steps, the parameters of Q' would be updated by the setting from Q.

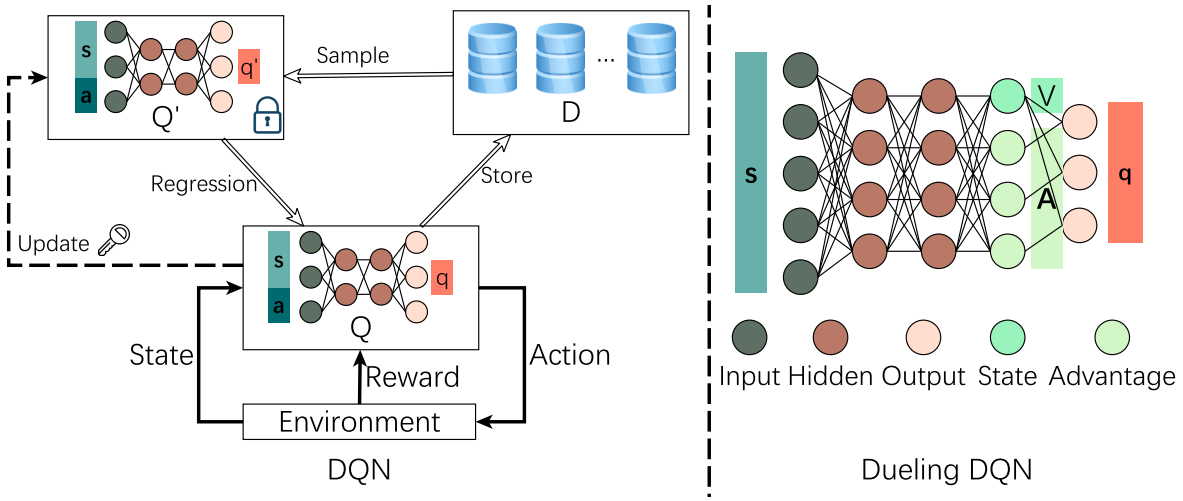

Figure 5. Deep Q Networks (DQN) and Dueling DQN.

In Formula (2), the target DNN Q' is responsible for both actions selection and evaluation. If any $Q'(ss, a')$ is overestimated, the entire evaluation is overestimated accordingly since we always tend to select the action $argmax_{a'}Q'(ss, a')$. To avoid this upward bias, Double DQN decouples the calculation operation by applying the training DNN Q to selection actions, while employing the target DNN Q' to evaluate actions. If Q is overestimated and select an improper action, Q' will rectify the bias. As shown in the right side of Figure 5, the Dueling DQN modifies the DNN structure by splitting the Q evaluation into two channels: a scalar V representing the contribution of state only and a vector A describing the reward from both state and action. It is worth noting that the expectation of A is limited to be 0. From the view of network training, the original DQN's output contains $|A|$ numbers with the range $[0, \infty]$ and now it switches to one scalar with range $[0, \infty]$ and $|A|$ numbers with zero expectation, which is more efficient to train. Moreover, both two outputs have practical meanings and can be used for further analysis. Another aspect of enhancing DQN performance is changing the uniform sample from memory pool D by considering their significance, which is called prioritized experience replay. More details can be found in Reference [54].

3.5.2. DNN as Actor (Policy-Based)

In policy-based method, DNN gets involved in the action selection directly instead of via $Q^{\pi}(s,a)$. The policy can be viewed as an optimization problem, where the objective function is maximizing reward expectation and the search space is policy space. The input of DNN is current state and output is the probability distribution of potential actions. By employing gradient ascent, we can update the DNN to provide better action then maximize total reward. Compared with value-based approaches, policy-based algorithms have better convergence and are more efficient

in dealing with high dimensional or continuous actions tasks. Additionally, policy-based algorithms can learn stochastic policies, while value-based schemes choose actions deterministically, like the action maximum Q value in Formula (2). On the other hand, the enormous solution space of policy makes it difficult to sample sufficiently, which results in large variance and local optimal solution.

3.5.3. Actor-Critic Model

Generally, compared with policy-based approach, the value-based method is less stable and suffer from poor convergence since the policy is derived based on $Q^{\pi}(s, a)$ approximation. But a value-based method is more sample efficient, while a policy-based method is easier to fall into local optimal solution because the search space is vast. The actor-critic model combines these two approaches, i.e., the agent contains two DNNs named actor and critic, respectively. In each training iteration, the actor considers current state s and policy π for deciding action a. Then, the environment changes to state s' and returns reward r. The critic updates its own parameters based on the feedback from the environment and output a mark for the actor's action. The actor updates policy π depending on the critic's mark. Some typical algorithms have been proposed recent years, like Deep Deterministic Policy Gradient (DDPG) [55] and Asynchronous Advantage Actor-Critic (A3C) [56], as Figure 6 illustrates.

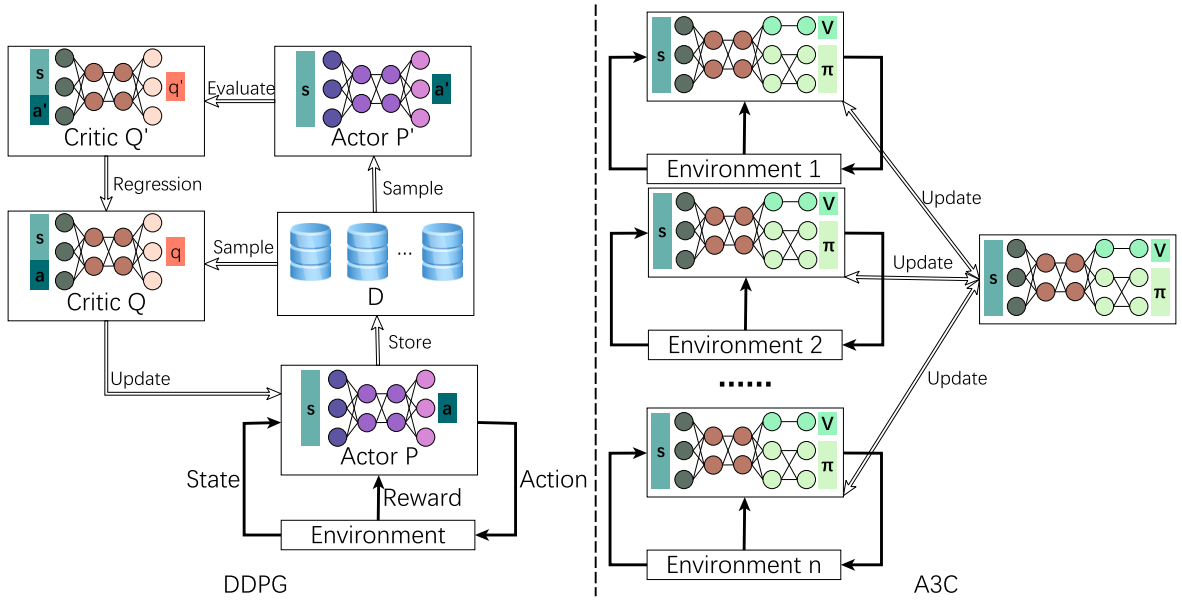

Figure 6. Deep Deterministic Policy Gradient (DDPG) and Asynchronous Advantage Actor-Critic (A3C).

Similar to Double DQN, DDPG also utilizes the idea of target DNN and experience replay memory. In DQN, the action is determined via trained DNN Q, while, in DDPG, the action is selected by actor DNN. Initially, target critic Q' and actor P' are the replicas of Q and P, respectively. In each training step, actor P plays with the environment and store its experience in-memory buffer D. The target actor P' samples batches of experiences randomly and selects action a'. Then, the target critic Q' evaluates the expected reward q' on action a'. Meanwhile, the trained critic Q also estimates the expected reward q but is based on the recorded action a. The critic Q and actor P are updated via square error and policy gradient, respectively, while the target network Q' and actor P' are also updated in a little progress.

In the scheme of A3C, the DNN structure is similar to dueling DQN. A3C does not apply experience replay memory but employ asynchronous training method. As shown in the right side of Figure 6, there are n global network's replicas interacting with their environments and each local DNN (including the actor and the critic) is trained independently. During the training episode, the local DNN does not update its own parameters but renovate the global. The local model synchronizes global DNN after several steps. So, the global network is the model we want and these independent local models accelerate the training process.

As a conclusion of this section, we summarize these illustrated DL schemes in Table 2 with their advantages and disadvantages, classified by their training methods and DNN architectures.

Table 2. Comparison of deep learning (DL) architectures.

Training Scenario	Architecture	Pros	Cons
Super--vised	FNN	•any closed and bounded function approximation •global spatial feature extraction	•model training and hyper parameter tuning •local spatial feature awareness
	CNN	•weight sharing to reduce computation complexity •local spatial feature awareness •mainstream deep model in computer vision, also be employed in speech and text processing	•some info missing due to downsampling •hyper parameter tuning •temporal-sequential task
	RNN: ESN	•fast training process •low dimensional temporal sequence processing •applied in speech processing, stock price prediction, language modeling, etc.	•hyper parameters tuning •high dimensional temporal sequence (like video) processing
	RNN: LSTM	•easing gradient vanishing •employed in sentiment analysis, machine translation, dialog, etc.	•computation complexity, not support parallel computing
	RNN:GRU	•easing gradient vanishing •faster training than LSTM	• not support parallel computing
Unsuper--vised	Auto Encoder	•feature extraction •data visualization	•training complexity
Reinfor--cement	Value-based: DQN	•general schemes, play different games	•the case of continuous actions •overestimation on reward values
	Value-based: Double DQN	•avoiding overestimation •faster convergence than DQN	•the case of continuous actions
	Value-based: Dueling DQN	•faster convergence than DQN and Double DQN •practical meaningful separation	•the case of continuous actions •structure complexity •less efficient on small state spaces
	Policy-based	•steady convergence •high dimensional and continuous tasks •stochastic policies learning	•large variance •local optima
	Actor-Critic: DDPG	•discrete and continuous tasks •steady convergence	•high complexity •slow training
	Actor-Critic: A3C	•efficient learning on continuous tasks •parallel training	•high complexity •intensive requirements on hardware

4. Deep Learning for Data Caching

We divide the studies regarding deep learning for data caching in edge networks into four categories depending on the DL tools employed: FNN and CNN, RNN, Auto Encoder, and DRL, as summarized in Tables 3–6, respectively. Recently, many works utilize more than one DL techniques for jointly considered caching problems. For instance, at the beginnin, we apply an RNN to predict content popularity and then a DRL to find suboptimal solutions of content placement for the purpose of reducing time complexity. In this case, we classify the related work into DRL since it represents the caching allocation policy. Unless there is mention of the caching location (such as CRs, MBSs, FBSs, EDs, and BBUs), the approaches in this section can be utilized for both Layer 1 and Layer 2 caching.

Table 3. Summary of fully-connected neural network (FNN) and convolutiuonal neural network (CNN) for data caching.

Study	Caching Problem	DL Objective	DL Architecture	Main Conclusions
[57]	content delivery	reduce feasible region of time slot allocation	CNN	The proposed method combine the prediction from CNN with the optimal branch & bound algorithm.
[58]	where to cache, content delivery	determine MBSs for caching & delivery duration	FNN	A well-trained FNN can achieve around 90% approximation to the optimum.
[59]	where to cache, content delivery	nominate proper CRs for caching	parallel CNN	The optimization model is transformed to a grayscale image and a cluster of CNNs are trained to capture spatial features.
[60]	where to cache, content delivery	reduce feasible region for caching	parallel CNN	The proposed framework provides a speed up by 75 times with an additional cost of less than 5% in specific cases.
[61]	what to cache	extract video features	3D CNN	Both published and unpublished videos are considered in the popularity predicting scheme.
[62]	what to cache	predict requested content & frequency	FNN	The historical visiting records in a metropolis-scale bus WiFi network help to predict future events.
[63]	what to cache	predict requested content	CNN	The users' interests are predicted by CNN analysis on tweets.
[64]	what to cache	predict content popularity	FNN	This paper questions the role of DNN in caching applications.

Table 4. Summary of recurrent neural network (RNN) for data caching.

Study	Caching Problem	DL Objective	DL Architecture	Main Conclusions
[65]	what to cache	predict requested content & user mobility	ESN	The proposed method combine the ESN framework with sublinear algorithms and the simulation is based on real data from content provider YouKu, as well as university BUPT.
[66]	what to cache	predict requested content & user mobility	ESN	A conceptor-based ESN approach is proposed for users' behavior prediction and this paper analyze the caching application at the level of UAVs.
[67]	what to cache	predict content popularity	CNN, Bidirectional LSTM& FNN	This paper tracks the association of temporal requests and achieves above 60% prediction accuracy.
[68]	what to cache	predict content popularity	GRU	The proposed framework combines RNN with the hedge strategy and the training data is from content provider iQiYi.
[69]	what to cache	predict content popularity	LSTM	The proposed scheme can learn the content popularity at long and short time scales. The experiments on iQiYi and Movielens dataset show it improves the hit ratio by 20∼32%.
[70]	what to cache	predict content popularity	LSTM	The paper proposes a LSTM model to predict content popularity and user mobility.
[71]	what to cache	predict content popularity	LSTM Seq2Seq	The content popularity prediction is recognized as a seq2seq model and a general framework for end-to-end cache making is created.
[72]	content delivery	reduce traffic load, select optimal BS subset	Auto Encoder& Bidirectional GRU Seq2Seq	A new coded caching approach using auto encoder to reduce network load and a learning model to solve the cover problem with attention scheme and beam search.

Table 5. Summary of auto encoder for data caching.

Study	Caching Problem	DL Objective	DL Architecture	Main Conclusions
[73]	what to cache	predict content popularity	stacked sparse auto encoder	An distributed auto encoder model is constructed by SDN/NVF technical to predict data packet popularity in WSN.
[74]	what to cache	predict content popularity	stacked sparse auto encoder	This paper proposes a method of deploying auto encoder and softmax distributed DNN in the 5G core network.
[75]	what to cache	predict content popularity	auto encoder & stacked denoising auto encoder	This article employs two auto encoders to extract hidden features of users and contents, which is utilized for further popularity estimation.
[76]	what to cache	predict content popularity	stacked auto encoder	This paper generates a distributed and reconfigurable prediction model via SDN.
[77]	what to cache	predict top popular contents	collaborative denoising auto encoder	An auto encoder model using a clustering method is proposed to improve the prediction accuracy and the performance is checked on ITRI and Netflix datasets.

Table 6. Summary of deep reinforcement learning (DRL) for data caching.

Study	Caching Problem	DL Objective	DL Architecture	Main Conclusions
[78]	what to cache	decide cache placement	prioritized experience replayed, dueling & RNN DQN	This paper consider a scenario where the popularity is dynamic and unknown in ultra dense network, from the view of energy efficiency.
[79]	what to cache	decide cache replacement	DDPG	The framework is constructed based on the Wolpertinger architecture and the results show it improves performance on both short-term and long-term.
[80]	what to cache	decide cache replacement	DDPG& multi-agent actor-critic scheme	This paper provides a DRL scheme for both centralized and decentralized caching, with the training of DDPG and temporal differences respectively.
[81]	what to cache	decide cache placement	DQN	This article proposes an online caching scheme for 360° videos.
[82]	what to cache	decide cache placement & service allocation	DQN	This paper pays attention to the issue of caching with QoE in edge-enabled IoT environment.
[83]	what to cache	decide cache placement	Q learning& DQN	An RL scheme for D2D-assisted cache-enabled IoT is proposed, where Q learning and DQN are utilized for user and SBS caching respectively.

Table 6. *Cont.*

Study	Caching Problem	DL Objective	DL Architecture	Main Conclusions
[84]	what to cache	decide cache placement	external memory-based DQN	The proposed algorithm can achieve an improvement in cache hit rate and long-term reward in a single BS scenario.
[85]	what to cache	decide cache placement	A3C	The IoT data transiency and dynamic context characteristics are jointly considered under data freshness and communication cost.
[86]	what to cache	decide cache placement	hyper DQN	This paper tackle a two-timescale caching task with employing DQNs in parallel.
[87]	what to cache	decide cache placement	DQN	A chunk-based caching scheme in data processing network considers network performance and processing efficiency.
[88]	what to cache	decide cache replacement& power allocation	DQN	This paper tries to minimize latency in a downlink of F-RAN in a centralized mode.
[89]	what to cache	predict popularity& model tuning	CNN, LSTM, Convolutional RNN& Q Learning	The DNN model is proposed to predict request demand and content popularity with Q learning for model selection.
[90]	where to cache	service allocation, computation offloading& cache placement	dueling DQN	The authors provide a social trust scheme with both direct and indirect observation in mobile social networks for edge computing, caching, and D2D.
[91]	content delivery	users grouping	dueling DQN	This paper considers both caching and interference alignment under time-varying channel.
[92]	where to cache &content delivery	BS connection, computation offloading& cache location	double dueling DQN	The authors propose an integrated scheme for dynamic orchestration of networking, caching and computing in vehicle networks.
[93]	what to cache, content delivery	decide caching & bandwidth allocation	DDPG	A cooperative caching policy among base station, roadside units and vehicles is proposed.
[94]	what to cache, content delivery	caching, computing offloading& radio allocation	actor-critic model with natural policy gradient	This article provides a joint method for caching, computing and radio allocation in the fog-enabled IoT to minimum average latency for all requests.
[95]	what to cache, content delivery	schedule multicast& replace cache	variational auto-encoder& double DQN	A double coded caching algorithm is proposed to increase the robustness of wireless transmission.
[96]	where & what to cache	predict popularity, decide caching & task offloading	GRU& multi-agent DQN	This paper proposes a joint caching and offloading algorithm in multi-user non-orthogonal multiple access mobile edge computing system.
[97]	where&what to cache, content delivery	predict user mobility& content popularity, D2D link	ESN, LSTM&actor-critic model	A joint content placement and delivery scheme is proposed to solve the medium access contention in the cache-enabled D2D network.

4.1. FNN and CNN

In Reference [57], the content delivery problem in wireless network is formulated as two MILP optimization models with the aims of minimum delivery time slot and energy consumption, respectively. Both models consider the data rate for content delivery. Considering the computational complexity of solving MILP, CNN is introduced to reduce the feasible region of decision variables, where the input is a channel coefficients matrix. The FNN in Reference [58] plays a similar role as in Reference [57] to simplify the searching space of the content delivery optimization model.

For resource allocation problems, the authors of Reference [98] model it as linear sum assignment problems then utilize CNN and FNN to solve the model. The idea is extended in Reference [59,60], where the authors consider where to cache problem among potential CRs and content delivery jointly, which is modeled as MILP for the sake of balancing caching and transmission cost by considering the user mobility, space utilization and bandwidth limitations. The cache allocation is viewed as a multi-label classification problem and is decomposed into several independent sub-problems, where each one correlates with a CNN to predict assignment. The input of CNN is a grey-scale image which combines the information of user mobility, space and link utilization level. In Reference [59], a hill-climbing local search algorithm is provided to improve the performance of CNN, while, in Reference [60], the prediction of CNN is used to feed a smaller MILP model.

For these above works [57–60,98], the FNN or CNN input is extracted from the optimization model. The work in Reference [99] trains a CNN via original graph instead of parameters matrix/image, which makes the process human recognizable and interpretable. Though the authors take the traveling salesman problem not data caching as an example, the method can be viewed as a potential research direction.

In Reference [61], an ILP model is proposed to minimize the backhaul video-data type load by determining the portion of cached content in BSs. Since the mobile users covered by a BS change frequently, therefore predicting user preference is unnecessary. Instead, the authors concentrate on popular content in general. In the beginning, a 3D CNN is introduced to extract spatio-temporal features of videos. The popularity of new contents without historical information is determined via comparing similar video features. The authors of Reference [62] also considers the spatio-temporal features among visiting contents in a mobile bus WiFI environment. By exploiting the previous 9 days collecting data, the content that the user may visit on the last day and corresponding visiting frequency can be forecast. The social property is taken into account in Reference [63]. By observing users interests on tweets during the 2016 U.S. election, a CNN based predicted model can foresee the content category that is most likely to be requested. Such kind of content would be cached in MBSs and FBSs.

The work of Reference [64] examines the role of DNN in caching from another aspect. The authors propose an FNN to predict content popularity as a regression problem. The results show that FNN outperforms RNN, though the later is believed to be effective to solve sequential predictions. Moreover, replacing the FNN by a linear estimator does not devalue the performance significantly. The author provides an explanation that FNN would work better than linear predictor in the case of incomplete information, and RNN has more advantages to model the popularity prediction as a classification rather than a regression problem.

4.2. RNN

Considering RNN is superior in dealing with sequential tasks, the work [67] applies a bidirectional RNN for online content popularity prediction in mobile edge network. Simple RNN's output depends on previous and current storage, but the bidirectional can also take future information into account. The forecast model consists of three blocks cascadingly: a CNN reads user requests and extracts features; bidirectional LTSM learns association of requests over time step; FNN is added in the end to improve the prediction performance. Then, content eviction is based on the popularity prediction.

The authors in Reference [65] utilize ESN to predict both content request distribution and end user mobility pattern. The user's preference is viewed as a context which links with personal information

combining gender, age, job, location, etc. For the request prediction, the input of ESN is user's information vector and the output represent the probability distribution of content. For mobility prediction, the input includes historical and present user's location and the output is the expected position for next time duration. Eventually, the prediction influences the caching content decisions in BBUs and RRHs for the purpose of minimizing traffic load and delay in CRAN. The authors extend their work in Reference [66] by introducing conceptor-based ESN which can split users' context into different patterns and learn them independently. Therefore, a more accurate prediction is achieved.

In Reference [68], a caching decision policy named PA-Cache is proposed to predict time-variant video popularity for cache eviction when the space is full. The temporal content popularity is exploited by attaching every hidden layer representation of RNN to an output regression. In order to improve the accuracy, hedge backpropagation is introduced during the training process which decides when and how to adapt the depth of the DNN in an evolving manner. Similarly, the work in Reference [69] also considers caching replacement of video content. A deep LSTM network is utilized for popularity prediction consisting of stacking multiple LSTM layers and one softmax layer, where the input of the network is request sequence data (device, timestamp, location, video's title) without any prepossessing and the output is estimated content popularity. Another work concentrates on prediction, and interactions between user mobility and content popularity can be found [70].

The work of Reference [71] recognizes the popularity prediction as a Seq2Seq modeling problem and proposes an LSTM Encoder-Decoder model. The input vector consists of past probabilities where each vector is calculated during a predefined time window. In Reference [72], the authors focus on caching content delivery with the aim of minimizing BSs to cover all requested users, i.e., set cover problem, via coded caching. Unlike Reference [71], an auto encoder is introduced in coded caching stage for file conversion to reduce transmission load. Additionally, an RNN model is employed to select BSs for broadcasting.

Reference [100] shows the potential of RNN in solving where to cache problem. In Reference [100], a task allocation model is formulated as a knapsack problem and the decision variables represent the task is processed locally in mobile devices (MDs) or remotely in edge servers (ESs). The authors design a multi-pointer network structure of 3 RNNs, where 2 encoders encode MDs and ESs, respectively, 1 decoder demonstrates ES and MD pairing. Considering the similarity of where to cache optimization model and knapsack problem, the multi-pointer network can be transferred for caching location decision after according parameter modifications.

4.3. Auto Encoder

Generally, the auto encoder is utilized to learn efficient representation or extract features of raw data in an unsupervised manner. The work in Reference [73] considers the cache replacement in wireless sensor network (WSN) based on content popularity. Considering sparse auto encoder (SAE) can extract representative expression of input data, the authors employ an SAE followed by a classifier where the input contains collecting user content requests and the output represents the contents popularity level. The authors also think about the implementation in a distributed way by SDN/NFV technical, i.e., the input layer is deployed on the sink node, while the rest layers are implemented on the main controller. A related work applying auto encoder in 5G network proactive caching can be found in Reference [74]. In Reference [75], two auto encoders are utilized for extracting the features of users and content, respectively. Then, the extracted information is explored to estimate popularity at the core network. Similarly, the auto encoder in Reference [76] is for spatio-temporal popularity features extraction and auto encoders work collaboratively in Reference [77] to predict top K popular videos.

4.4. DRL

In Reference [78], the authors concentrate on the influence of caching strategies on energy consumption of ultra dense network in the scenario where the popularity is dynamic and unknown. In order to associate contents and cache positions, a DRL algorithm with prioritized experience replay,

dueling structure, as well as RNN, is proposed. Simulation indicates it is good at both stationary and dynamic popularity scenarios. The authors in Reference [81] study the problem of caching 360° videos and virtual viewports in FBSs with unknown content popularity. The virtual viewport represents the most popular tiles of a 360° video over users' population. A DQN is introduced to decide which tiles of a video to be hosted and in which quality. Additionally, Reference [82] employs DQN for content eviction decision offering a satisfactory quality of experience, and Reference [83] is for the purpose of minimizing energy consumption. In Reference [84], the authors also apply DQN to decide cache eviction in a single BS. Moreover, the critic is generated with stacking LSTM and FNN to evaluate Q value and an external memory is added for recording learned knowledge. To improve the prediction accuracy, the Q value update is determined by the similarity of the estimated value of critic and recording information in the external memory, instead of critic domination. Reference [86] puts forth DQN a two-level network caching, where a parent node links with multiple leaf nodes to cache content instead of a single BS. In Reference [79], a DRL framework with Wolpertinger architecture [101] is presented for content caching at BSs. The Wolpertinger architecture is based on actor-critic model and performs efficiently in large discrete action space. Reference [79] employs two FNNs working as actor and critic, respectively, where the former determines requested content is cached or not and the later estimates the reward. The whole framework consists of two phases: in the offline phase, these two FNNs are trained in supervised learning; in the online phase, the critic and actor update via the interaction with environment. The authors extend their work to a multi agent actor-critic model for decentralized cooperative caching at multiple BSs [80]. In Reference [85], an actor-critic model is used for solving the cache replacement problem, which balances the data freshness and communication cost. The aforementioned papers put attention on network performance while they ignore the influence of caching on information processing and resource consumption. Therefore, authors of Reference [87] design cache policy considering both network performance during content transmission and processing efficiency during data consumption. A DQN is employed to determine the number of chunks of the requested file to be updated. During the training process, the authors in Reference [87] also compare the performance between dueling DQN and original DQN. According to their simulations, dueling DQN does not outperform the original DQN but consumes more time to convergence, which contradicts the cognition that dueling DQN accelerates the training [8]. One proper reason is the action space is not large enough as shown in paper [53]. The paper [88] investigates a joint cache replacement and power allocation optimization problem to minimize latency in a downlink F-RAN. A DQN is proposed for finding a suboptimal solution. Though Reference [89] is regarded as solving what to cache problem, like Reference [79–88,102], the reinforcement learning approach plays a different role. In Reference [89], a DNN is utilized for content popularity prediction and then an RL is used for DNN hyperparameters tuning instead of determining caching content. Therefore, the action space consists of choosing model architectures (i.e., CNN, LSTM, etc.), number of layers, and layer configurations.

In Reference [90], the authors generate an optimization model with the aim of maximizing network operator's utility in mobile social networks under the framework of mobile edge computing, in-network caching and D2D communications (3C). The trust value which if estimated through social relationships among users are also considered. Then, a DQN model is utilized for solving the optimization problem, including determining video provider and subscriber association, video transcoding offloading, and the video cache allocation for video providers. The DQN employs two CNNs for the training process, where one generates target Q value and the other is for estimated Q value. Unlike the conventional DQN, the authors in Reference [90] introduces a dueling DQN structure, which helps achieve a more robust result. The authors also consider utilizing dueling DQN model in different scenarios, like cache-enabled opportunistic interference alignment [91] and orchestrating 3C in vehicular network [92]. Reference [93] provides a DDPG model to cope with continuous valued control decision for 3C in vehicular edge networks. The work in Reference [102] focuses on the cooperative caching policy at FBSs with maximum distance separable coding in ultra

dense networks. A Q-learning model is utilized to determine caching categories and the content quantity at FBSs during the off-peak duration. The authors also consider the performance of their algorithm on continuous action cases because of action discretization. Compared with DDPG, there is only a tiny performance gap in small-scale case. A very interesting result is that in large-scale case, the proposed Q-learning model even slightly outperforms DDPG due to underfitting, which express the complicated network structure of DDPG.

Reference [94] provides an optimization model which takes what to cache and content delivery into consideration in the fog-enabled IoT network in order to minimize service latency. Since the wireless signals and user requests are stochastic, an actor-critic model is engaged where the actor makes decisions for requesting contents, while the critic estimates the reward. Especially, the action space S consists of decision variables and reward function is a variant of the objective function. In order to avoid converging to local optima, the nature policy gradient method is employed in Reference [94]. A caching replacement strategy and dynamic multicast scheduling strategy are studied in Reference [95]. In order to get a suboptimal result, an auto encoder is used to approximate the state. Further, a weighted double DQN scheme is utilized for avoiding overestimation of Q value. Reference [96] applies an RNN to predict content popularity by collecting historical requests and the output represents the popularity in the near future. Then, the prediction is employed for cooperative caching and computation offloading among MEC servers, which is modeled as an ILP problem. For the purpose of solving it efficiently, a multi-agent DQN is applied where each user is viewed as an agent. The action space consists of task local computing and offloading decision, as well as local caching and cooperative caching determination. The reward is measured by accumulated latency. The agent chooses its own action based on current state without cooperation. The where to cache, what to cache, and content delivery decision of D2D network are jointly modeled in Reference [97]. Two RNNs, ESN and LSTM, are considered to predict mobile users' location and requested content popularity. Then, the prediction result is used for determining content categories and cache locations. The content delivery is formulated as the actor-critic based DRL framework. The state spaces include CSI, transmission distances, and communication power between the requested user and other available candidates. The function of DRL is determining the communication link among users with the aim of minimizing power consumption and content delay.

We notice that most papers prefer to use value-based model (critic) and value-policy-based (actor-critic) model in DRL framework, but rare paper considers only the policy-based model to solve data caching problem. One proper reason is that the search space of caching problem is enormous so the policy-based model is easier to fall into local optimal solution, resulting in poor performance. Though the value-based model is less stable, some variant structures are utilized, like Double DQN, in Reference [95], to avoid value overestimation and dueling DQN, in Reference [90–92], to improve robustness.

5. Research Challenges and Future Directions

A serious of open issues on content caching and potential research directions are discussed in this section. We first extend the idea of content caching to virtual network function chain since caching can be viewed as a specific network function. Then, we consider the caching for augmented reality applications. Moreover, we notice that the cache dimensioning has not been covered yet by DL methods. Finally, we debate the addition cost introduced by DL.

5.1. Caching as a Virtual Network Function Chain

The concept of Network Function Virtualization (NFV) has been firstly discussed and proposed within the realms of the European Telecommunication Standardization Institute (ETSI) (Network Functions Virtualization, An Introduction, Benefits, Enablers, Challenges, and Call for Action, ETSI, 2012 https://portal.etsi.org/NFV/NFV_White_Paper.pdf). The rationale is to facilitate the dynamic provisioning of network services through virtualization technologies to decouple the service creation

process form the underlying hardware. The framework allows network services to be implemented by specific chaining and ordering of a set of functions which can be implemented either on a more traditional dedicated hardware, which, in this case, are called Physical Network Functions (PNFs), alternatively, as Virtual Network Functions (VNFs), which is a software running on top of virtualized general-purpose hardware. The decoupling between the hardware and the software is one of the important considerations the other—equally important-—is that a virtualized service lend itself naturally to a dynamic programmable service creation where VNF resources can be deployed as required. Hence, edge cloud and network resource usage can be adapted to the instantaneous user demand whilst avoiding more static over-provisioned configurations.

Within that framework, the incoming network service requests include the specification of the service function chain that need to be created in the form of an ordered sequence of VNFs. For example, different types of VNFs, such as a firewall or a NAT mechanism, need to be visited in a specific order. In such a constructed service chain, each independent VNF requires specific underlying resources in terms for example of CPU cycles and/or memory.

Under this framework, caching of popular content can be considered as a specialized VNF chain function since inevitably delivery of the cached popular content to users will require a set of other functions to be supported related to security, optimization of the content etc. However, the issue of data caching and VNF chaining have evolved rather independently in the literature and the issue on how to optimize data caching when seeing it as part of VNF chain is still an interesting open ended issue. In this case, the aforementioned DL architectures can be applied to this augmented problem where in addition to caching, the associated VNF issues need to be taken into account, such as VNF chaining, routing, and hosting.

5.2. Caching for Mobile Augmented Reality (MAR) Applications and Digital Twins (DTs)

Mobile augmented reality (MAR) applications can be considered as a way to augment the physical real-world environment with artificial computer-based generated information and is an area that has received significant research attention recently. In order to successfully superimpose different digital object in the physical world, MAR applications include several computationally and storage complex concepts, such as image recognition, mobile camera calibration, and also the use of advanced 2D and 3D graphics rendering. These functionalities are highly computationally intensive and as such require support from an edge cloud, in addition the virtual objects to be embedded in the physical world are expected to be proactively cached closer to the end user so that latency is minimized. Ultra low latency in these type of applications is of paramount importance so that to provide a photorealistic embedding of virtual objects in the video view of the end user. However, since computational and augmented reality objects need to be readily available, the caching of those objects should be considered in conjunction with the computational capabilities of the edge cloud. In addition to the above, when MAR is considered under the lenses of an NFV environment, the application might inherently require access to some VNFs; therefore, the above discussion on VNF chaining for MAR applications is also valid, in this case.

Recently, the concept of Digital Twin (DT) [103,104] has received significant research attention due to the plethora of applications ranging from industrial manufacturing and health to smart cities. In a nutshell, a DT can be defined as an accurate digital replica of a real world object across multiple granularity levels; and this real world object could be a machine, a robot or an industrial process or (sub) system. By reflecting the physical status of the system under consideration in a virtual space open up a plethora of optimization, prediction, fault tolerance and automation process that cannot be done using solely the physical object. At the core of DT applications is the requirement of stringent two–way real time communication between the digital replica and the physical object. This requirement inevitably requires support from edge clouds to minimize latency and efficient storage and computational resources, including caching. In that setting, the use of the aforementioned deep learning technologies will have a key role to play in order to provide high quality real time

decision making to avoid misalignment between the digital replica of the physical object under consideration. Efficient machine-to-DT connectivity would require capabilities similar to the above mentioned augmented reality application, but, due to the continuous real-time control-loop operation, DTs will require a completely new set of network optimization capabilities, and, in that frontier, efficient caching and data-driven techniques will have a central role to play. Hence, as the research regarding the inter-play between low latency communications and DTs is still in the embryonic stage, there is significant scope in the investigation of suitable data-driven deep learning techniques to be utilized for distributed allocation of caching and computing resources.

5.3. Deep Learning for Cache Dimensioning

As introduced in Section 2, cache dimensioning explores the appropriate cache size allocation for content hosts, such as CRs and BSs. Disappointingly, there is rarely a paper that applies DL on cache dimensioning decisions. One proper reason is lack of training data set in contrast to content popular prediction, where we have historical user request log to train a DNN. In addition, the caching size allocation affects the network performance and economic investment. Recently, network slicing has been identified as an important tool to enable 5G to provide multi-services with diverse characteristics. The slice is established on physical infrastructure including network storage. Therefore, it is a very interesting topic to consider the allocation of the memory space to support content caching and other storage services, which guarantees QoE and satisfies task requirements. Furthermore, for the case lacking a training data set, DRL can be viewed as a promising technology to configure slicing settings, as well as cache dimensioning. For the action space designing, it can be either discrete by setting storage levels, or continuous which allocates the memory space directly. However, there are requirements to design a caching-enabled network slicing model especially for dynamic allocation, as well as associated DRL framework, including state space, detailed action space, reward function, and agent structure.

5.4. The Cost of Deep Learning

Though the application of DL brings performance efficiency for caching policy, additional cost introduced by DL is unneglected, since training and deploying the DL model require not only network resources but also time duration. Naturally, there is a trade-off between the cost which DL-assisted caching policy saved and the consumption which supports DL itself running, which indicates the trading with DL results in either profit, loss, or break even. Therefore, where and when to apply DL should be carefully investigated. This is especially the case for many deep learning frameworks that utilize specialized processing cores in the form of Graphics processing units (GPUs) and for some special scenarios, like DL on maritime IoT [105] and terrestrial satellite systems [106]. Moreover, not only the DL architecture but also the implementation approach (i.e., centralized or distributed deployment) would affect the balance between the cost saving and consuming, where the distributed method can be supported by end user devices equipped with DL processing chip [107]. Besides, for the purpose of reducing resource consumption and accelerating training process, some knowledge transfer methods, like transfer learning [108], can be utilized, which can transform the knowledge already learnt from the source domain to a relevant target domain.

6. Conclusions

This article presents a comprehensive study for the application of deep learning methods in the area of content caching. Particularly, the data caching assignment is divided into two classifications according to the caching location of edge network. Each category contains where to cache, what to cache, cache dimensioning, and content delivery. Then, we introduce typical DNN methods, which are categorized via training process into supervised learning, unsupervised learning, and RL. Further, this paper critically compares and analyzes state-of-the-art papers on parameters, such as DL methods employed, the caching problems solved, and the objective of applying DL.

The challenges and research directions of DL on caching are also examined on the topic of extending caching to VNF chains, the application of caching for MAR, as well as DTs, DL for cache size allocation, and the additional cost of employing DL. Undoubtedly, DL is playing a significant role in 5G and beyond. We hope this paper will increase discussions and interests on DL for caching policy design and relevant applications, which will advance future network communications.

Author Contributions: Conceptualization, Y.W.; Writing–original draft, Y.W.; Writing–review & editing, V.F.; Supervision, V.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, X.; Chen, M.; Han, Z.; Wu, D.O.; Kwon, T.T. TOSS: Traffic offloading by social network service-based opportunistic sharing in mobile social networks. In Proceedings of the IEEE INFOCOM 2014—IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014; pp. 2346–2354.
2. Sze, V.; Chen, Y.H.; Yang, T.J.; Emer, J.S. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* **2017**, *105*, 2295–2329. [CrossRef]
3. Sun, Y.; Peng, M.; Zhou, Y.; Huang, Y.; Mao, S. Application of machine learning in wireless networks: Key techniques and open issues. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3072–3108. [CrossRef]
4. Kulkarni, A.; Seetharam, A. Model and Machine Learning based Caching and Routing Algorithms for Cache-enabled Networks. *arXiv* **2020**, arXiv:2004.06787.
5. Shuja, J.; Bilal, K.; Alanazi, E.; Alasmary, W.; Alashaikh, A. Applying Machine Learning Techniques for Caching in Edge Networks: A Comprehensive Survey. *arXiv* **2020**, arXiv:2006.16864.
6. ANOKYE, S.; Mohammed, S.; Guolin, S. A Survey on Machine Learning Based Proactive Caching. *ZTE Commun.* **2020**, *17*, 46–55.
7. Chen, M.; Challita, U.; Saad, W.; Yin, C.; Debbah, M. Artificial neural networks-based machine learning for wireless networks: A tutorial. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3039–3071. [CrossRef]
8. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3133–3174. [CrossRef]
9. Wang, X.; Han, Y.; Leung, V.C.; Niyato, D.; Yan, X.; Chen, X. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 869–904. [CrossRef]
10. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
11. Peng, M.; Sun, Y.; Li, X.; Mao, Z.; Wang, C. Recent advances in cloud radio access networks: System architectures, key techniques, and open issues. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2282–2308. [CrossRef]
12. Paschos, G.S.; Iosifidis, G.; Tao, M.; Towsley, D.; Caire, G. The role of caching in future communication systems and networks. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1111–1125. [CrossRef]
13. Shan, S.; Feng, C.; Zhang, T.; Loo, J. Proactive caching placement for arbitrary topology with multi-hop forwarding in ICN. *IEEE Access* **2019**, *7*, 149117–149131. [CrossRef]
14. Wang, Y.; Zheng, G.; Friderikos, V. Proactive caching in mobile networks with delay guarantees. In Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
15. Fang, C.; Yu, F.R.; Huang, T.; Liu, J.; Liu, Y. An energy-efficient distributed in-network caching scheme for green content-centric networks. *Comput. Netw.* **2015**, *78*, 119–129. [CrossRef]
16. Sahoo, J.; Salahuddin, M.A.; Glitho, R.; Elbiaze, H.; Ajib, W. A survey on replica server placement algorithms for content delivery networks. *IEEE Commun. Surv. Tutor.* **2016**, *19*, 1002–1026. [CrossRef]
17. Kabir, A.; Rehman, G.; Gilani, S.M.; Kitindi, E.J.; Ul Abidin Jaffri, Z.; Abbasi, K.M. The role of caching in next generation cellular networks: A survey and research outlook. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3702. [CrossRef]

18. Traverso, S.; Ahmed, M.; Garetto, M.; Giaccone, P.; Leonardi, E.; Niccolini, S. Temporal locality in today's content caching: Why it matters and how to model it. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 5–12. [CrossRef]
19. Dabirmoghaddam, A.; Barijough, M.M.; Garcia-Luna-Aceves, J. Understanding optimal caching and opportunistic caching at "the edge" of information-centric networks. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 47–56.
20. Shi, Y.; Larson, M.; Hanjalic, A. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv. (CSUR)* **2014**, *47*, 1–45. [CrossRef]
21. Jung, J.; Berger, A.W.; Balakrishnan, H. Modeling TTL-based Internet caches. In Proceedings of the IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428), San Francisco, CA, USA, 30 March–3 April 2003; Volume 1, pp. 417–426.
22. Fofack, N.C.; Nain, P.; Neglia, G.; Towsley, D. Performance evaluation of hierarchical TTL-based cache networks. *Comput. Netw.* **2014**, *65*, 212–231. [CrossRef]
23. Rossi, D.; Rossini, G. On sizing CCN content stores by exploiting topological information. In Proceedings of the 2012 Proceedings IEEE INFOCOM Workshops, Orlando, FL, USA, 25–30 March 2012; pp. 280–285.
24. Xu, Y.; Li, Y.; Lin, T.; Wang, Z.; Niu, W.; Tang, H.; Ci, S. A novel cache size optimization scheme based on manifold learning in content centric networking. *J. Netw. Comput. Appl.* **2014**, *37*, 273–281. [CrossRef]
25. Paschos, G.; Iosifidis, G.; Caire, G. Cache optimization models and algorithms. *arXiv* **2019**, arXiv:1912.12339.
26. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*; Association for Computing Machinery: New York, NY, USA, 2009; pp. 1–12.
27. Wang, L.; Bayhan, S.; Ott, J.; Kangasharju, J.; Crowcroft, J. Understanding scoped-flooding for content discovery and caching in content networks. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1887–1900. [CrossRef]
28. Li, L.; Zhao, G.; Blum, R.S. A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1710–1732. [CrossRef]
29. Gharaibeh, A.; Khreishah, A.; Ji, B.; Ayyash, M. A provably efficient online collaborative caching algorithm for multicell-coordinated systems. *IEEE Trans. Mob. Comput.* **2015**, *15*, 1863–1876. [CrossRef]
30. Golrezaei, N.; Molisch, A.F.; Dimakis, A.G.; Caire, G. Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution. *IEEE Commun. Mag.* **2013**, *51*, 142–149. [CrossRef]
31. Afshang, M.; Dhillon, H.S.; Chong, P.H.J. Fundamentals of cluster-centric content placement in cache-enabled device-to-device networks. *IEEE Trans. Commun.* **2016**, *64*, 2511–2526. [CrossRef]
32. Chen, Z.; Liu, Y.; Zhou, B.; Tao, M. Caching incentive design in wireless D2D networks: A Stackelberg game approach. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6.
33. Ye, Z.; Pan, C.; Zhu, H.; Wang, J. Tradeoff caching strategy of the outage probability and fronthaul usage in a cloud-RAN. *IEEE Trans. Veh. Technol.* **2018**, *67*, 6383–6397. [CrossRef]
34. Ren, D.; Gui, X.; Zhang, K.; Wu, J. Mobility-Aware Traffic Offloading via Cooperative Coded Edge Caching. *IEEE Access* **2020**, *8*, 43427–43442. [CrossRef]
35. Song, J.; Choi, W. Mobility-aware content placement for device-to-device caching systems. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 3658–3668. [CrossRef]
36. Liu, D.; Chen, B.; Yang, C.; Molisch, A.F. Caching at the wireless edge: Design aspects, challenges, and future directions. *IEEE Commun. Mag.* **2016**, *54*, 22–28. [CrossRef]
37. Peng, X.; Zhang, J.; Song, S.; Letaief, K.B. Cache size allocation in backhaul limited wireless networks. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6.
38. Liu, A.; Lau, V.K. How much cache is needed to achieve linear capacity scaling in backhaul-limited dense wireless networks? *IEEE/ACM Trans. Netw.* **2016**, *25*, 179–188. [CrossRef]

39. Song, J.; Choi, W. Minimum cache size and backhaul capacity for cache-enabled small cell networks. *IEEE Wirel. Commun. Lett.* **2017**, *7*, 490–493. [CrossRef]
40. Zhou, B.; Cui, Y.; Tao, M. Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks. *IEEE Trans. Commun.* **2017**, *65*, 2956–2970. [CrossRef]
41. Maddah-Ali, M.A.; Niesen, U. Fundamental limits of caching. *IEEE Trans. Inf. Theory* **2014**, *60*, 2856–2867. [CrossRef]
42. Ha, V.N.; Le, L.B. Coordinated multipoint transmission design for cloud-RANs with limited fronthaul capacity constraints. *IEEE Trans. Veh. Technol.* **2015**, *65*, 7432–7447. [CrossRef]
43. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 12 October 2020).
44. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
45. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
46. Siegelmann, H.T.; Sontag, E.D. Turing computability with neural nets. *Appl. Math. Lett.* **1991**, *4*, 77–80. [CrossRef]
47. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
48. Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2015; pp. 2692–2700.
49. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional sequence to sequence learning. *arXiv* **2017**, arXiv:1705.03122.
50. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
51. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
52. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
53. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*; PMLR: New York, NY, USA, 2016; pp. 1995–2003.
54. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
55. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
56. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*; PMLR: New York, NY, USA, 2016; pp. 1928–1937.
57. Lei, L.; Yuan, Y.; Vu, T.X.; Chatzinotas, S.; Ottersten, B. Learning-Based Resource Allocation: Efficient Content Delivery Enabled by Convolutional Neural Network. In Proceedings of the 2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Cannes, France, 2–5 July 2019; pp. 1–5.
58. Lei, L.; You, L.; Dai, G.; Vu, T.X.; Yuan, D.; Chatzinotas, S. A deep learning approach for optimizing content delivering in cache-enabled HetNet. In Proceedings of the 2017 International Symposium on Wireless Communication Systems (ISWCS), Bologna, Italy, 28–31 August 2017; pp. 449–453.
59. Wang, Y.; Friderikos, V. Caching as an image characterization problem using deep convolutional neural networks. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
60. Wang, Y.; Friderikos, V. Network Orchestration in Mobile Networks via a Synergy of Model-driven and AI-based Techniques. *arXiv* **2020**, arXiv:2004.00660.

61. Doan, K.N.; Van Nguyen, T.; Quek, T.Q.; Shin, H. Content-aware proactive caching for backhaul offloading in cellular network. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 3128–3140. [CrossRef]
62. Qin, Z.; Xian, Y.; Zhang, D. A neural networks based caching scheme for mobile edge networks. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 408–409.
63. Tsai, K.C.; Wang, L.; Han, Z. Mobile social media networks caching with convolutional neural network. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Barcelona, Spain, 15–18 April 2018; pp. 83–88.
64. Fedchenko, V.; Neglia, G.; Ribeiro, B. Feedforward Neural Networks for Caching: N Enough or Too Much? *ACM SIGMETRICS Perform. Eval. Rev.* **2019**, *46*, 139–142. [CrossRef]
65. Chen, M.; Saad, W.; Yin, C.; Debbah, M. Echo state networks for proactive caching in cloud-based radio access networks with mobile users. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 3520–3535. [CrossRef]
66. Chen, M.; Mozaffari, M.; Saad, W.; Yin, C.; Debbah, M.; Hong, C.S. Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 1046–1061. [CrossRef]
67. Ale, L.; Zhang, N.; Wu, H.; Chen, D.; Han, T. Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network. *IEEE Internet Things J.* **2019**, *6*, 5520–5530. [CrossRef]
68. Fan, Q.; Li, J.; Li, X.; He, Q.; Fu, S.; Wang, S. PA-Cache: Learning-based Popularity-Aware Content Caching in Edge Networks. *arXiv* **2020**, arXiv:2002.08805.
69. Zhang, C.; Pang, H.; Liu, J.; Tang, S.; Zhang, R.; Wang, D.; Sun, L. Toward edge-assisted video content intelligent caching with long short-term memory learning. *IEEE Access* **2019**, *7*, 152832–152846. [CrossRef]
70. Mou, H.; Liu, Y.; Wang, L. LSTM for Mobility Based Content Popularity Prediction in Wireless Caching Networks. In Proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
71. Narayanan, A.; Verma, S.; Ramadan, E.; Babaie, P.; Zhang, Z.L. Deepcache: A deep learning based framework for content caching. In *Proceedings of the 2018 Workshop on Network Meets AI & ML*; Association for Computing Machinery: New York, NY, USA, 2018; pp. 48–53.
72. Zhang, Z.; Zheng, Y.; Li, C.; Huang, Y.; Yang, L. On the Cover Problem for Coded Caching in Wireless Networks via Deep Neural Network. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
73. Lei, F.; Cai, J.; Dai, Q.; Zhao, H. Deep learning based proactive caching for effective wsn-enabled vision applications. *Complexity* **2019**, *2019*, 5498606. [CrossRef]
74. Lei, F.; Dai, Q.; Cai, J.; Zhao, H.; Liu, X.; Liu, Y. A proactive caching strategy based on deep Learning in EPC of 5G. In *Proceedings of the International Conference on Brain Inspired Cognitive Systems*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 738–747.
75. Rathore, S.; Ryu, J.H.; Sharma, P.K.; Park, J.H. DeepCachNet: A proactive caching framework based on deep learning in cellular networks. *IEEE Netw.* **2019**, *33*, 130–138. [CrossRef]
76. Liu, W.X.; Zhang, J.; Liang, Z.W.; Peng, L.X.; Cai, J. Content popularity prediction and caching for ICN: A deep learning approach with SDN. *IEEE Access* **2017**, *6*, 5075–5089. [CrossRef]
77. Lin, Y.T.; Yen, C.C.; Wang, J.S. Video Popularity Prediction: An Autoencoder Approach With Clustering. *IEEE Access* **2020**, *8*, 129285–129299. [CrossRef]
78. Li, W.; Wang, J.; Zhang, G.; Li, L.; Dang, Z.; Li, S. A reinforcement learning based smart cache strategy for cache-aided ultra-dense network. *IEEE Access* **2019**, *7*, 39390–39401. [CrossRef]
79. Zhong, C.; Gursoy, M.C.; Velipasalar, S. A deep reinforcement learning-based framework for content caching. In Proceedings of the 2018 52nd Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 21–23 March 2018; pp. 1–6.
80. Zhong, C.; Gursoy, M.C.; Velipasalar, S. Deep Reinforcement Learning-Based Edge Caching in Wireless Networks. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *6*, 48–61. [CrossRef]
81. Maniotis, P.; Thomos, N. Viewport-Aware Deep Reinforcement Learning Approach for 360 Video Caching. *arXiv* **2020**, arXiv:2003.08473.
82. He, X.; Wang, K.; Xu, W. QoE-driven content-centric caching with deep reinforcement learning in edge-enabled IoT. *IEEE Comput. Intell. Mag.* **2019**, *14*, 12–20. [CrossRef]

83. Tang, J.; Tang, H.; Zhang, X.; Cumanan, K.; Chen, G.; Wong, K.K.; Chambers, J.A. Energy minimization in D2D-assisted cache-enabled internet of things: A deep reinforcement learning approach. *IEEE Trans. Ind. Inform.* **2019**, *16*, 5412–5423. [CrossRef]
84. Wu, P.; Li, J.; Shi, L.; Ding, M.; Cai, K.; Yang, F. Dynamic content update for wireless edge caching via deep reinforcement learning. *IEEE Commun. Lett.* **2019**, *23*, 1773–1777. [CrossRef]
85. Zhu, H.; Cao, Y.; Wei, X.; Wang, W.; Jiang, T.; Jin, S. Caching transient data for Internet of Things: A deep reinforcement learning approach. *IEEE Internet Things J.* **2018**, *6*, 2074–2083. [CrossRef]
86. Sadeghi, A.; Wang, G.; Giannakis, G.B. Deep reinforcement learning for adaptive caching in hierarchical content delivery networks. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 1024–1033. [CrossRef]
87. Wang, Y.; Li, Y.; Lan, T.; Aggarwal, V. Deepchunk: Deep q-learning for chunk-based caching in wireless data processing networks. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 1034–1045. [CrossRef]
88. Rahman, G.S.; Peng, M.; Yan, S.; Dang, T. Learning based joint cache and power allocation in fog radio access networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4401–4411. [CrossRef]
89. Thar, K.; Oo, T.Z.; Tun, Y.K.; Kim, K.T.; Hong, C.S. A deep learning model generation framework for virtualized multi-access edge cache management. *IEEE Access* **2019**, *7*, 62734–62749. [CrossRef]
90. He, Y.; Liang, C.; Yu, R.; Han, Z. Trust-based social networks with computing, caching and communications: A deep reinforcement learning approach. *IEEE Trans. Netw. Sci. Eng.* **2018**. [CrossRef]
91. He, Y.; Zhang, Z.; Yu, F.R.; Zhao, N.; Yin, H.; Leung, V.C.; Zhang, Y. Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks. *IEEE Trans. Veh. Technol.* **2017**, *66*, 10433–10445. [CrossRef]
92. He, Y.; Zhao, N.; Yin, H. Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach. *IEEE Trans. Veh. Technol.* **2017**, *67*, 44–55. [CrossRef]
93. Qiao, G.; Leng, S.; Maharjan, S.; Zhang, Y.; Ansari, N. Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks. *IEEE Internet Things J.* **2019**, *7*, 247–257. [CrossRef]
94. Wei, Y.; Yu, F.R.; Song, M.; Han, Z. Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning. *IEEE Internet Things J.* **2018**, *6*, 2061–2073. [CrossRef]
95. Zhang, Z.; Chen, H.; Hua, M.; Li, C.; Huang, Y.; Yang, L. Double coded caching in ultra dense networks: Caching and multicast scheduling via deep reinforcement learning. *IEEE Trans. Commun.* **2019**, *68*, 1071–1086. [CrossRef]
96. Li, S.; Li, B.; Zhao, W. Joint Optimization of Caching and Computation in Multi-Server NOMA-MEC System via Reinforcement Learning. *IEEE Access* **2020**, *8*, 112762–112771. [CrossRef]
97. Li, L.; Xu, Y.; Yin, J.; Liang, W.; Li, X.; Chen, W.; Han, Z. Deep reinforcement learning approaches for content caching in cache-enabled D2D networks. *IEEE Internet Things J.* **2019**, *7*, 544–557. [CrossRef]
98. Lee, M.; Xiong, Y.; Yu, G.; Li, G.Y. Deep neural networks for linear sum assignment problems. *IEEE Wirel. Commun. Lett.* **2018**, *7*, 962–965. [CrossRef]
99. Ling, Z.; Tao, X.; Zhang, Y.; Chen, X. Solving Optimization Problems Through Fully Convolutional Networks: An Application to the Traveling Salesman Problem. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**. [CrossRef]
100. Jiang, Q.; Zhang, Y.; Yan, J. Neural Combinatorial Optimization for Energy-Efficient Offloading in Mobile Edge Computing. *IEEE Access* **2020**, *8*, 35077–35089. [CrossRef]
101. Dulac-Arnold, G.; Evans, R.; van Hasselt, H.; Sunehag, P.; Lillicrap, T.; Hunt, J.; Mann, T.; Weber, T.; Degris, T.; Coppin, B. Deep reinforcement learning in large discrete action spaces. *arXiv* **2015**, arXiv:1512.07679.
102. Gao, S.; Dong, P.; Pan, Z.; Li, G.Y. Reinforcement learning based cooperative coded caching under dynamic popularities in ultra-dense networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5442–5456. [CrossRef]
103. El Saddik, A. Digital Twins: The Convergence of Multimedia Technologies. *IEEE MultiMedia* **2018**, *25*, 87–92. [CrossRef]
104. Alam, K.M.; El Saddik, A. C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems. *IEEE Access* **2017**, *5*, 2050–2062. [CrossRef]
105. Huo, Y.; Dong, X.; Beatty, S. Cellular Communications in Ocean Waves for Maritime Internet of Things. *IEEE Internet Things J.* **2020**. [CrossRef]

106. Gao, J.; Zhao, L.; Shen, X. Service offloading in terrestrial-satellite systems: User preference and network utility. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
107. Huo, Y.; Dong, X.; Xu, W.; Yuen, M. Enabling multi-functional 5G and beyond user equipment: A survey and tutorial. *IEEE Access* **2019**, *7*, 116975–117008. [CrossRef]
108. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]