Received 20 April 2022; accepted 10 May 2022. Date of publication 17 May 2022; date of current version 25 May 2022.

Digital Object Identifier 10.1109/OJCOMS.2022.3175927

Adapting Deep Learning for Content Caching Frameworks in Device-to-Device Environments

RAHUL BAJPAI, SOURADEEP CHAKRABORTY, AND NAVEEN GUPTA

Department of Electrical and Electronics Engineering, BITS Pilani (K.K. Birla Goa Campus), Goa 403726, India

CORRESPONDING AUTHOR: R. BAJPAI (e-mail: p20190003@goa.bits-pilani.ac.in)

ABSTRACT Recently, we have witnessed an expeditious growth in the intelligence and processing ability of user equipment accompanied by the explosive increase of wireless data and traffic. With such a huge demand, and a shortage of resources to fulfill that need, device-to-device (D2D) communication has surfaced as a propitious solution to reduce costs associated with backhaul links by collaboratively caching files. This paper explores the existing content caching frameworks in D2D communication environments. Further, it proposes two novel frameworks - one leveraging a combination of recurrent and deep neural networks and another based on the latest deep language models that use attention mechanisms known as transformers. The developed frameworks require minimum apriori knowledge about the environment and utilize the growth of user data to achieve performance enhancement. Our experiments show that the proposed frameworks are adaptive in nature and learn from the historical data of the environment. Further, we achieve an overall 25% increase in the D2D cache hit rate over a recently proposed framework that uses neural networks for collaborative filtering (NCF) to make caching decisions.

INDEX TERMS Traffic offloading, D2D caching, machine learning, deep learning, edge intelligence, time-series.

I. INTRODUCTION

With the exceptional growth of cellular data traffic and to pave the way for sixth-generation (6G) communication systems, mobile user equipment (UE) like smartphones need to perform complex computations within micro-seconds [1]. These needs are being accomplished by the introduction of cheaper yet powerful Internet of Things (IoT) devices capable of leveraging emerging technologies such as machine learning (ML) and deep learning (DL) for resource allocation and data retrieval in communication systems [2]. Simultaneously, since the number of users has grown exponentially, data retrieval is a huge bottleneck to rank performance gains [3]. Due to this recent phenomenal increase in the number of users and their processing power and storage, device-to-device (D2D) communications have emerged as a solution to this convoluted challenge.

In D2D communication systems, single or multiple users can communicate directly with each other by leveraging superior processing capabilities without routing their data through the main base station (BS), thereby significantly reducing backhaul link costs and transmission delays [2]. Thus, D2D systems have been a highly demanding area of research in the development of the fifth-generation (5G) and beyond cellular networks. The success of D2D communications depends on the ability of the system to make decisions on which files are needed to be cached by a D2D user to achieve optimum performance.

Caching is the action of storing frequently accessed popular files in temporary storage. Preliminary research on caching in wireless networks shows that the backhaul link (connection with the BS) remains underutilized during off-peak times and congested during peak traffic [4], [5]. These backhaul links can be utilized by designing small-sized cells such as femtocell and microcell with small base stations (SBSs) having the capability of prefetching and caching the frequently accessed popular files during off-peak times. These frameworks are referred to as heterogeneous networks (HetNets). The HetNets are capable of solving the problem of underutilization of off-peak times; however, they suffered from unreliable, low-rate, relatively slow links due to the increasing number of edge-users [6].

In caching-enabled D2D environments, users are able to cache popular content for proximity users and themselves (i.e., users with low communication costs). Users can effectively cache popular files and transmit them via direct D2D links upon request [7]. Whenever the content is requested by a D2D user, it first checks its local cache for data, and if local cache hit is unavailable, then caches of the D2D users in proximity will be checked for the content. A backhaul link to the BS only gets established if there are no D2D cache hits as well [7]. In this way, D2D communications facilitate faster content delivery with reduced latency [2].

Caching-enabled D2D communications can relieve the network burden by leveraging backhaul links by collaboratively caching files. However, the caching policy design becomes challenging to replicate in real-world conditions due to evolving content popularity trends, uncertainty of user mobility patterns, and limited content storage capacities. In previous works, these important factors are assumed to be well known or ignored completely. For example, in [8] the content popularity was assumed to be following a Zipf distribution, while in the user preferences and user behavior was assumed to be constant. In [9], the authors show that if these factors are all taken into account, then finding an optimal solution becomes mathematically intractable.

In this paper, we leverage DL to formulate caching strategies to determine which files are needed to be cached by a user to optimize the system's performance. Popular baseline strategies such as the least recently used (LRU) and least frequently used (LFU) depend on impractical presumptions [10] such as adversarial or constant content requesting models. These initial caching policies provide an insignificant cache-hit ratio (CHR) whenever the content popularity distribution is different from these presumptions [11]. For instance, in a D2D system, the content popularity can be differentiated and concentrated based on region, time, or both. Further, authors in [10], and [11] fail to consider user mobility and temporal relations (like requesting the same file every Monday at the office). The subsequent section discussed related works that try to solve these problems, both with and without DL, and highlighted the significant issues faced in designing the D2D caching strategies.

A. RELATED LITERATURE

There is a plethora of related literature available on wireless caching, and the engrossed readers can refer [12], and [13] for a concise survey related explicitly to D2D caching frameworks considering the mobility of the users. Many contemporary works rely on the use of parametric models for their caching frameworks. For instance, authors in [14] use the theory of variations for capturing dynamic popularity. Further, in [15], the authors use random content replacements in the file catalog for mobile access networks. These parametric models require careful fine-tuning before they can be applied to real-world data, and hence, require large execution times [16]. Thus, these computationally complex models are unsuitable for real-time scenarios, where aspects of the environment like content popularity, user preferences, and locations are fast-changing.

Another field that is witnessing a spotlight in recent years is ML and DL. Due to several advancements in these techniques, the user's preferences, locations, and content popularities can be learned and predicted over time [17]. The success of various applications of these technologies depends on the problem formulation for the learning models. For instance, in [18], instead of learning the uneven content popularity, the user's content access has been learned to improve the performance using ML. Additionally, these learning techniques achieve improved performance when intelligent environmental inferences are integrated into the caching policy design. The authors of [19] assumed that the social relationship is also essential for designing caching strategies due to privacy concerns. Further, a caching policy is designed using neural network collaborative filtering (NCF), integrating social relationships with DL and recommendation algorithms in D2D scenarios. However, the mobility of the user has not been incorporated in [18] and [19].

Due to advancements in ML technology, the recent emerging algorithms are capable of predicting random events with considerable accuracy. Recurrent neural networks (RNNs) have emerged as one such model that is capable of handling and predicting time-series data such as user mobility. In general, most of the existing works uses DL or traditional deep reinforcement learning (DRL) techniques. In [20], the prediction of the content popularity and the mobility of users have been discussed utilizing RNNs. The simulation results demonstrated that combining mobility with content predictions leads to significant performance gains in cache hits. However, [20] uses deep reinforcement learning (DRL), and the reliance on powerful computing is significantly higher, contrary to deep neural networks (DNNs). Further, DNNs are faster to converge and provide a lightweight solution to optimize the caching decision (CD).

Using a single model is not feasible for mobile network environments with large state spaces. As such, authors in [21] achieved higher cache-hit ratios by separating the modeling of content popularity and user mobility with the CD. CDs in [21] are made utilizing the Long Short-Term Memory (LSTM) networks assembled with Omniscale convolutional neural network (CNN). The proposed R2-D2D framework does not require any prior information, such as the distribution of file popularity or assumptions like the environment's stationarity. However, the R2-D2D framework presented in [21] is limited to the combination of RNN and DNN, and there is a need to explore the latest deep language models that use attention mechanisms known as transformers for CDs.

Motivated by the existing literature, this paper formulates the caching problem in the following manner: we take advantage of the latest innovations in DL, like RNNs, using a non-stationary system model. In addition, we explore the incremental benefit of adding complexity to the system by introducing recent techniques like the attention framework. Specifically, for a D2D enabled environment, we have proposed two CD frameworks: the ‘RNN’ and the ‘attention’ framework. The RNN framework uses RNNs in conjunction with DNNs to make the CD. On the other hand, the attention framework uses a special transformer architecture for the same. To the best of our knowledge, this is the first time the attention framework has been designed to work with time-series inputs for D2D caching, unlike traditional language models, which use a corpus of natural language data [22].

B. MAJOR CONTRIBUTIONS

We have summarized our major contributions below:

- *A non-stationary model:* We designed a system model considering user mobility and dynamic file access patterns. User file requests have been explicitly defined along with D2D communications cost to determine the inherent randomness in the file popularities and the user’s mobility, respectively.
- *Stochastic time-series model:* LSTM networks have been implemented to predict the D2D communication costs and user file requests for individual users over time. Thus, the stochastic evolution of time-series variables depicting user mobilities and file popularities has been efficiently modeled utilizing RNNs.
- *Using the Attention mechanism:* A D2D content caching mechanism using an attention framework for time-series prediction has been developed. This framework scales well with real-life scenarios making minimal assumptions, leaving scope for exploring any future developments in DL.
- *Comparative Performance Analysis:* Through extensive experimentation, we quantitatively compare the proposed RNN and attention frameworks with each other, and also with related works. Results show that our developed caching policies are able to achieve state of the art performance by leveraging DL.

II. METHODOLOGY

A. SYSTEM MODEL

As shown in Fig. 1, the system model consists of M mobile users m_i, $(i \in 1, 2, \ldots, M)$ with limited cache storage, distributed randomly within a circular cell governed by a BS. Cache storage corresponding to mobile user m_i is denoted by S_i where $S_i \leq S_{BS}$, and S_{BS} is equal to the total storage needed for storing all the files by the BS. It is considered that BS has all the files, whereas cellular users can request and cache some of these files. Any library F consists of a total of n files[1] denoted as $f_j (j \in 1, 2, \ldots, n)$. All other important symbols are listed in Table 1. Without loss of generality, it is considered that each user is able to cache, receive or share at least one file at each timestep. Partial caching cases and multi-hop routing have not been considered to reduce system complexity.

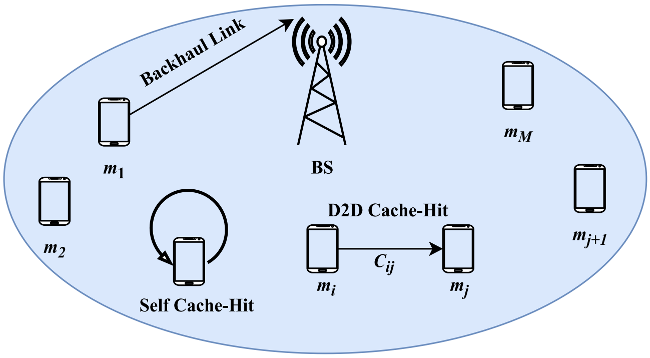

FIGURE 1. System model: states after making content requests.

TABLE 1. List of symbols.

Symbol	Description
m_i	i^{th} mobile user
f_j	j^{th} content file
$c_{i,j}$	Connection cost between i^{th} and j^{th} mobile user
c_{cutoff}	Cutoff connection cost in the D2D system
S_{BS}	Cache size of BS
S_i	Cache size of i^{th} user
ρ_t	Random variable to add stochasticty to cost
N_i	Set of k D2D neighbours of i^{th} user
R_{i,f_j}	1 if f_j is requested by m_i
$\theta_{f_j}(m_i)$	1 if f_j is cached by m_i
CHR_{D2D}	Ratio of file requests served by D2D cache-hits

The user’s mobility and their spatial locations are modeled using a cost-based approach. Every D2D connection between two proximity users is associated with a cost $c_{i,j} \in [0, 1]$, where $c_{i,j}$ is directly proportionate to the geographical distance between mobile users m_i and m_j. The cost is then modeled stochastically, considering that the cost would also be governed by physical layer channel conditions such as data requirement, power, and bandwidth. When a user requests a file, there are three possibilities. Either the file already exists in the user’s local cache or is available to a nearby user. If none of these conditions are satisfied, the user uses the backhaul link to fulfill this request through the BS itself. To incorporate this, we use a *cutoff* cost, c_{cutoff}, such that D2D connections are only established between user m_i and m_j only if $c_{i,j} \leq c_{cutoff} \leq c_{i,BS}$.

The non-stationarity in the environment is introduced to the costs at each timestep when they are updated as

$$\left(c_{i,j}\right)_{t+1} = \min\left\{\max\left(\left[\left(c_{i,j}\right)_t + \rho_t\right], 0\right), 1\right\}, \tag{1}$$

and

$$\left(c_{i,BS}\right)_{t+1} = \min\left\{\max\left(\left[\left(c_{i,BS}\right)_t + \rho_t\right], 0\right), 1\right\}, \tag{2}$$

where, ρ_t is a uniformly distributed random variable defined as, $\rho_t \sim U(-0.1, 0.1)$. In this manner, we restrict the costs from receiving impractical updates in every timestep; precisely, a maximum of 10% shifts are permitted.

[1] Please note that all files are considered uniform with unit size.

Thus, based on the c_{cutoff}, each user m_i will have k D2D neighbors defined as

$$N_i = \{m_a, m_b, \ldots, m_k\} \mid \forall m_j \in N_i, c_{i,j} \leq c_{cutoff}, \quad (3)$$

where, $k \leq M$, and relies completely on the costs between m_i and other mobile users. As a result, D2D connections for m_i is established only if the minimal communication cost of m_i with its neighbours is below $c_{i,BS}$, i.e., $c_{i,BS} > \min(c_{i,j})$, $\forall\ j$ such that user $m_j \in N_k$.

D2D users can request for the files represented by the file request variable $R_{i,f_j} \in \{0, 1\}$ on each timestep. R_{i,f_j} is defined as,

$$R_{i,f_j} = \begin{cases} 1 & \text{if user } i \text{ requests file } f_j, \\ 0 & \text{if user } i \text{ does not request file } f_j. \end{cases}$$

Hence, depending upon whether the i^{th} user has requested the j^{th} file, every mobile user creates a vector R_i consisting of n elements, and the j^{th} element is denoted as R_{i,f_j}.

At each timestep, an arbitrary dynamic probability distribution is considered for the file popularity. The following two frameworks are considered to model the file popularity. In the first framework, we employ a two-stacked network consisting of an LSTM layer and a DNN layer (the RNN framework), and in another, we employ a concurrent series of encoder and decoder networks using the attention mechanism popularized in transformers [22].

In the RNN framework, we use an LSTM network for modeling the file popularity to predict access patterns, while another LSTM predicts and models the costs. By utilizing the above, the current environmental state (costs) and the access pattern can be predicted for the CD. The CD is enabled by a DNN; however, this two-stack framework adds a level of complexity. This motivates us to consider an attention framework, where historical information from the past is used to make the CD. Specifically, historical data such as files requested by each user and the connection costs at each timestep are used to make the CD. The CD policy of these frameworks is summarized in the next section.

The outputs of both frameworks are a caching placement variable, $\theta_{f_j}(m_i)$, defined for each of the M users as

$$\theta_{f_j}(m_i) = \begin{cases} 1 & \text{if user } i \text{ caches file } f_j, \\ 0 & \text{if user } i \text{ does not caches file } f_j. \end{cases}$$

The CD is the process of populating the vector $\theta(m_i)$, with N elements. The j^{th} element in this vector, $\theta_{f_j}(m_i)$, represents whether the i^{th} user will cache the j^{th} file or not.

B. PROBLEM FORMULATION

In D2D communication, the goal is to reduce the usage of the backhaul link as much as possible. It can be achieved by maximizing the overall average D2D cache-hit ratio (CHR_{D2D}), which gets updated whenever a user's file request gets addressed by a D2D neighbor.[2] Thus, the CHR_{D2D} can be defined as:

$$CHR_{D2D} = \frac{\sum_{i=1}^{n} \sum_{f_j \in F} R_{i,f_j} \times \left(1 - \theta_{f_j}(U_i)\right) \times \Theta_{i,j}}{R}, \quad (4)$$

where, R denotes the total number of content requests, and $\Theta_{i,j}$ =min($\sum_{m_k \in N_i} \theta_{f_j}(U_k)$, 1). If no neighboring users of m_i consist of the j^{th} file in their cache storage, $\Theta_{i,j} = 0$. Here, the neighbors of each mobile user are determined and clustered depending on cutoff costs, and every mobile user requests a single file only. The optimization problem can be formulated as:

$$\text{maximize} \quad CHR_{D2D} \quad (5)$$

$$\text{subject to} \quad \sum_{f_j \in F} \theta_{f_j}(m_i) \leq S_i, \quad (5a)$$

$$\sum_{i=1}^{M} \theta_{f_j}(m_i) \leq M, \quad (5b)$$

$$\sum_{f_j \in F} R_{i,f_j} = 1. \quad (5c)$$

Constraint (5a) prohibits the required storage from exceeding the available cache storage for that user for caching files, and constraint (5b) prevents a user from caching the identical content multiple times. Further, constraint (5c) denotes that a mobile user can request only a single file at every timestep. The optimization problem formulated in (5) is NP-hard [23], and finding an optimal solution is mathematically intractable. A heuristic procedure using the predictive kind of DL algorithms is proposed for obtaining a suboptimal solution.

III. CACHING DECISION

A. THE RNN FRAMEWORK

1) RNNS FOR AUGMENTING CACHING EFFECTIVENESS

In RNNs, the output of each layer is used in conjunction with the inputs from the next timestep or batch, thereby implementing a memory of the historical data, which has been widely shown to be effective for forecasting [24]. Thus, RNNs are used to model and forecast the time-series data representing the environment. However, the RNN experiences gradient vanishing problems for complex long-term datasets. So, standard RNN frameworks are not utilized for modeling time-series data such as linkage costs or content requests. LSTM is a specific RNN that is efficiently utilized for time-series modeling to resolve the issues encountered by standard RNNs [24].

This paper utilizes LSTMs for predicting mobile user's file access patterns (R_{i,f_j}) and their connection costs ($c_{i,j}$). Initially, the costs are predicted, and subsequently, these values are utilized as additional input for predicting the user's file requests. The LSTM primarily consists of four gates - forget, input, update, and output gate. The following equations govern the values of these four gates:

$$f_t = \sigma\left(W_f^x \cdot x_t + W_f^h \cdot h_{t-1} + b_f\right), \quad (6)$$

[2] It is noteworthy that this metric differs from the local cache-hit ratio, which would be updated when user fulfills its file request using its own local cache.

Algorithm 1 Cost Prediction Using LSTM

```
Inputs: Time-series of historical cost data
    $D = (c_{i,j})_0, (c_{i,j})_1, ..., (c_{i,j})_{t-1}$ for all user pairs
Outputs: Cost prediction at next timestep $(\hat{\rho}_{i,j})_t$
Merge inputs to create feature set- features
Selecting the size of training window - timesteps
for n − epochs and batchsize do
    Input sampling and organizing into $D$ with shape [batchsize, timestep, features]
    Organizing $D$ into $D_{train}$ and $D_{test}$
    Training network $L$ utilizing batch of $D_{train}$
end for
Optimize: Loss based on MSE between $(\hat{c}_{i,j})_t$ and $(c_{i,j})_t$
Running predictions utilizing $L$ on $D_{test}$
```

Algorithm 2 User's File Request Prediction Using LSTM

```
Inputs: $(\hat{c}_{i,j})_t$ and $(R_i)_0, (R_i)_1, ...(R_i)_{t-1}$ for each user
Outputs: $(\hat{R}_i)_t$
Merge inputs to create feature set- features
Selecting the size of training window - timesteps
for n − epochs and batchsize do
    Input sampling and organizing into $D$ with shape [batchsize, timestep, features]
    Organizing $D$ into $D_{train}$ and $D_{test}$
    Training network $L$ utilizing batch of $D_{train}$
    Optimize: Loss dependent on MSE between $(R_i)_t$ and $(\hat{R}_i)_t$
end for
Running predictions utilizing $L$ on $D_{test}$
```

$$i_t = \sigma\left(W_i^x \cdot x_t + W_i^h \cdot h_{t-1} + b_i\right), \tag{7}$$

$$C_t = f_t * C_{t-1} + i_t * \left(\tanh\left(W_c^x \cdot x_t + W_c^h \cdot h_{t-1} + b_c\right)\right), \tag{8}$$

$$h_t = \tanh(C_t) * \left(\sigma\left(W_o^x \cdot x_t + W_o^h \cdot h_{t-1} + b_o\right)\right). \tag{9}$$

Here, x_t is input vector, and f_t, i_t, C_t, h_t denotes the forget, input, cell update and output gate values, respectively. W_p^q denotes the weight matrix of the connection layer between the vector q and the gate p, where, $p \in \{f, i, c, o\}$ and $q \in \{x, h\}$ corresponding to x_t or previous output vector (h_{t-1}). The tanh and σ are the standard activation functions. The LSTMs for prediction of $(c_{i,j})_t$ and user's file request are shown in Algorithm 1 and 2, respectively.

2) DNNS FOR OPTIMIZING CACHING DECISION

DNNs are able to treat the LSTM outputs as a feature, learning how to correct for errors in the predictions of user requests and costs with time. On receiving the LSTM predictions (specific to each particular UE in the environment), the DNN decides which files should be cached at that timestep, i.e., it generates the cache placement vector $\theta(m_i)$ for each user m_i. The users then schedule their actual file requests, and the CHR_{D2D} is evaluated after all users generate their file request vector R_i, as stated earlier in (4).

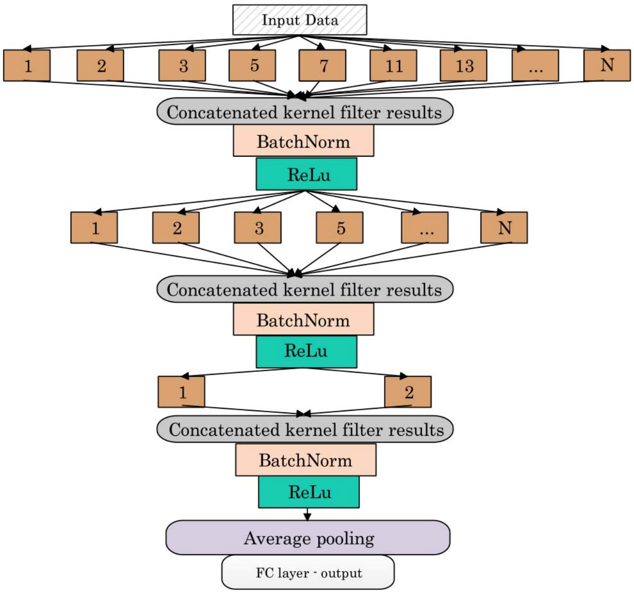

FIGURE 2. DNN Model Architecture - Omni-Scale 1D-CNN.

The back propagation mechanism in the DNN optimizes the CHR_{D2D} by representing the CD as a classification problem. There are four input features engineered for the DNN. The historical CDs and the CHR associated with each decision are the two time-series inputs, and the predicted file requests $(\hat{R}_i)_t$ and connection costs $(\hat{\rho}_{i,j})_t$ are the UE-specific time-series inputs. These predictions get generated by the preceding LSTM layers and are then used by the DNN. The outputs are the cache placement vector for all mobile users, i.e., $\theta(m_i) = [\theta_1(m_i), \theta_2(m_i), \ldots, \theta_{f_j}(m_i)], \ldots, \theta_{f_n}(m_i)]$. As there are a total of M users in the environment, M such vectors will be generated.

The DNN's architecture follows a novel 1-dimensional CNN (1D-CNN) having multi-scale kernel sizes designed specifically for time-series data [25]. Fig. 2 shows a DNN model architecture comprising three convolution layers having kernels of prime number sizes (1, 2, 3, 5, 7, 11, 13,. . . , N). The global average pooling layer is used for the feature extraction, while a fully connected layer acts as the final classifier that decides on the files to be cached. Here, N represents the largest kernel size, specified as the largest prime number in such a way that $N < l/4$, where l is the training data length. Additionally, the hyperparameters are selected following the standard values utilized in [25] to show the validity of the model. As a result, a simpler model is adopted to minimize complexity while maintaining parsimony.

B. THE ATTENTION FRAMEWORK

1) TRANSFORMERS AND ATTENTION

RNN-based methods struggle to find long-term dependencies in data. To alleviate this, the transformer has been recently proposed for sequence modeling and obtained major success. Various recent works such as [1], [24]–[26] have applied it to music, speech, translation, and image generation. To

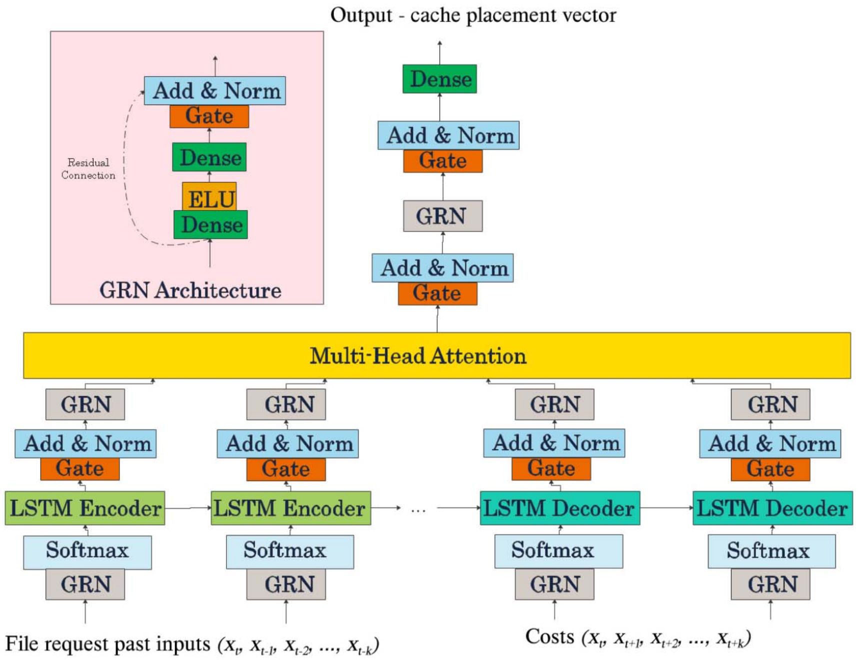

FIGURE 3. Attention Framework - Model Architecture.

find the temporal relation, the transformers can be used to access the historical information for all distances, to make it potentially more appropriate for grasping the repetitive patterns with long-term dependencies.

The transformer architecture utilizes the multi-head self-attention mechanism, enabling the transformer to seize both short and long-term dependencies. Moreover, various attention heads discover to concentrate on different features of temporal patterns. These prospective benefits make the transformer suitable for time series forecasting, like predicting which files will be requested next by a user based on its previous file access patterns. We have briefly introduced transformer architecture here.[3]

In the transformer's self-attention layer, a sub-layer known as multi-head self-attention, converts Y into H different query matrices $Q_h = Y.W_h^Q$, key matrices $K_h = Y.W_h^K$, and value matrices $V_h = Y.W_h^V$, respectively, where, $h = \{1, 2, \dots, H\}$. Here, $W_h^V \in R^{(d+1)d_v}$, and $W_h^Q, W_h^K \in R^{(d+1)d_k}$ are learning parameters. Following these linear projections, the scaled dot-product attention evaluates a vector outputs sequence given as:

$$O_h = \text{Attention}(Q_h, K_h, V_h) = \text{softmax}\Big(\big(Q_h.K_h^T\big)/\sqrt{d_k}.M\Big) \times V_h. \qquad (10)$$

A mask matrix M is used to prevent future information leakage that removes rightward attention by making all upper triangular elements to $-\infty$. Subsequently, $O_1, O_2, \dots, O_H$ are linked and linearly projected. A position-wise feed-forward sub-layer consisting of two layers of a fully connected network and a rectified linear unit (ReLU) activation in the middle is stacked upon the attention output.

2) MODEL DESCRIPTION

The model architecture of the attention framework is shown in Fig. 3. We use gating mechanisms at every timestep of historical information (comprising of past file requests and costs). As training progresses, these gated residual networks (GRNs) learn which timesteps are more important than others. It is done by the temporal multi-head attention, which learns both long-term and short-term relationships within the data. The GRNs comprise of a dense layer activated using the exponential linear unit (ELU) function followed by a gate which is passed to a standard layer normalization. The gates provide the flexibility to suppress any parts of the architecture that are not required for a certain timestep [26].

To learn long-term and short-term relationships across different timesteps, we use the multi-head attention in transformer-based architectures [22], [27]. The attention mechanism uses key (K) value (V) pairs along with queries (Q) in the following way:

$$\text{MultiHead}(Q, K_h, V_h) = \big[H_1, H_2, \dots, H_{m_H}\big].W_H, \qquad (11)$$

$$H_h = \text{Attention}\Big(Q_h.W_h^Q, K_h.W_h^K, V_h.W_h^V\Big), \qquad (12)$$

where, W_h^Q, W_h^K, W_h^V are head-specific weights for queries, keys and values.

The entire architecture uses the same optimization goals as the DNN in the RNN framework. The loss function minimizes the cache-miss, or $1 - CHR_{D2D}$, for all users jointly.

[3] A comprehensive introduction can be found in [22].

TABLE 2. Simulation parameters.

Parameters	Values
Number of Users (M)	500
Number of files (n)	200
Cache size of UEs (S_i)	50
Cache size of BS (S_{BS})	200
Cutoff-cost (c_{cutoff})	0.3
Macrocell Radius	150 m

Thus, the loss function is defined as follows:

$$\lambda(\Omega, W) = \sum_{y_t \in \Omega} \sum_{\tau=1}^{\tau_{\max}} L\big(y_t, \hat{y}(t-\tau, \tau)/M_{\tau_{\max}}, \qquad (13)$$

$$L(y, \hat{y}) = q\big(y - \hat{y}\big). \qquad (14)$$

Here, Ω represents the training data containing a total of M samples, W represents the weights of the layers, and q is the standard Adam optimizer used widely in DL.

IV. PERFORMANCE EVALUATION

A. EXPERIMENTAL SETTINGS

Various experimental settings are considered by changing the number of mobile users, cache storage of each user, and samples in the training set. As a base case, we have considered that 500 D2D users are arbitrarily distributed inside a small cellular cell of 150m radius and a library of 200 files of unit size. As stated in (1), we have considered the D2D communication costs to be governed by factors like fading, channel noise, user's location, and at each timestep, the costs are revised randomly. The c_{cutoff} has been taken as 0.3 to ensure significant D2D neighbors at each timestep.

For the comparison of our approaches quantitatively with others, we use two fundamental performance metrics - the traffic offloading and the overall average D2D cache hit rate. The traffic offloading is calculated by dividing the content items (in bytes) delivered by the local cache or D2D cache by the content items (in bytes) requested. Alternatively, the traffic offloading is the addition of local cache hit rate (CHR_L) and CHR_{D2D}, hence

$$\text{Traffic offloading} = CHR_{D2D} + CHR_L, \qquad (15)$$

and,

$$CHR_L = \frac{\sum_{i=1}^{M} \sum_{f_j \in F} R_{i,f_j} * \big(\theta_{f_j}(U_i)\big)}{R}. \qquad (16)$$

The rationale behind separating the traffic offloading from the CHR_{D2D} is that we need to see to what extent the proposed framework incentivizes D2D cache hits.

B. RESULTS AND DISCUSSIONS

This section showcases the performances of the proposed frameworks against several other popular approaches. These comprise the popularly used LFU and LRU algorithms, a recent NCF framework (implementation following the works of [19]), the RNN framework, represented as "R2-D2D", (as given in [21]) and a logistic regression (LR), a standard ML classification algorithm based framework. We also perform an ablation analysis for our RNN framework by considering the two stacks separately. First, the vRNN, where the CD is solely made based on file request predictions of the LSTM layer, and then the vDNN, where CD is made solely by the DNN layer using historical inputs instead of LSTM predictions. The attention framework utilizes the novel transformer mechanism as described above. The simulation parameters are listed in Table 2.

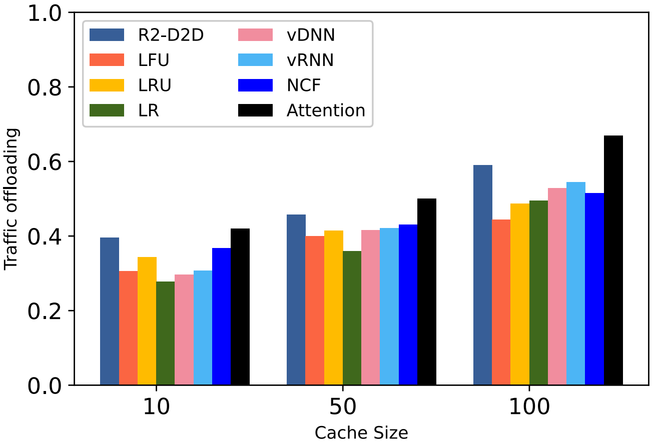

FIGURE 4. Traffic offloading - varying cache size.

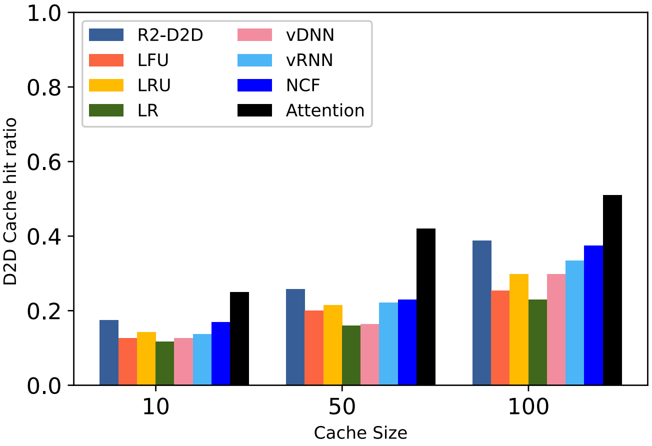

FIGURE 5. CHR_{D2D} - varying cache size.

The results show that the attention framework outperforms all others across all settings. Fig. 4 and Fig. 5 show the traffic offloading, and D2D cache hit ratio performance for the three cache sizes 10, 50, and 100, keeping the training set size fixed at 500 simulations. We find that both traffic offloading and CHR_{D2D} increases with cache size, which is in line with our expectations: as the cache size increases, more content can be cached by the UEs. We can observe that when the cache size is 100, the attention and RNN frameworks attain approximately 0.6 traffic offloading, outperforming the NCF and LRU frameworks by almost 15% and 25%, respectively. The RNN and attention approaches show a steeper improvement in performance with an increase in cache size

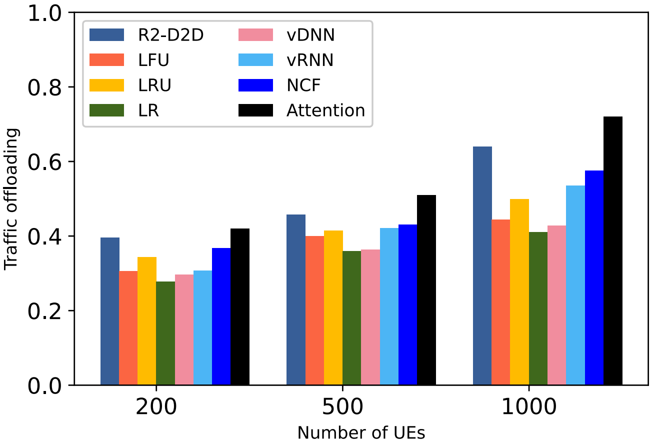

FIGURE 6. Traffic offloading - varying number of UEs.

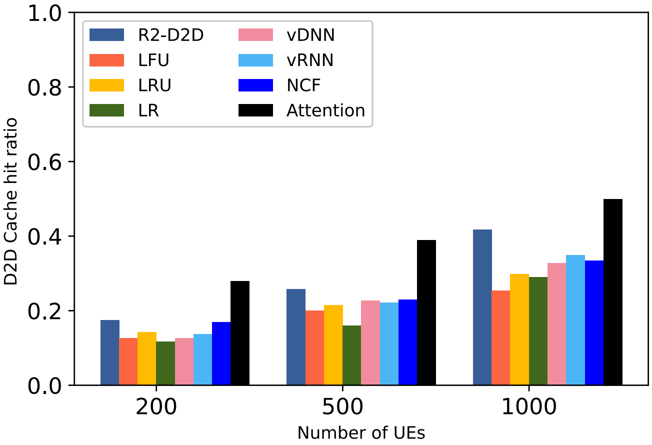

FIGURE 7. CHR_{D2D} - varying number of UEs.

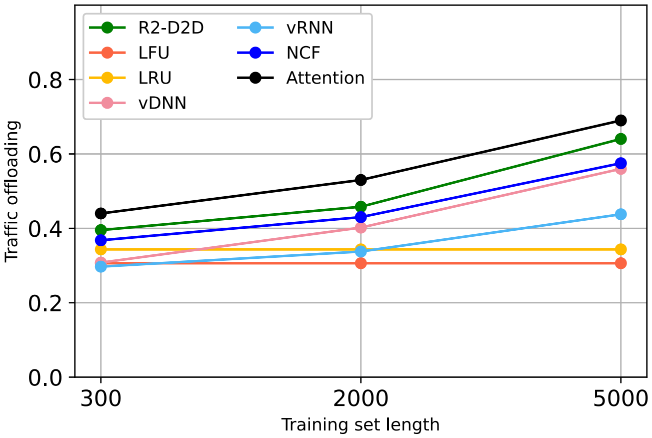

FIGURE 8. Traffic offloading - varying training set size.

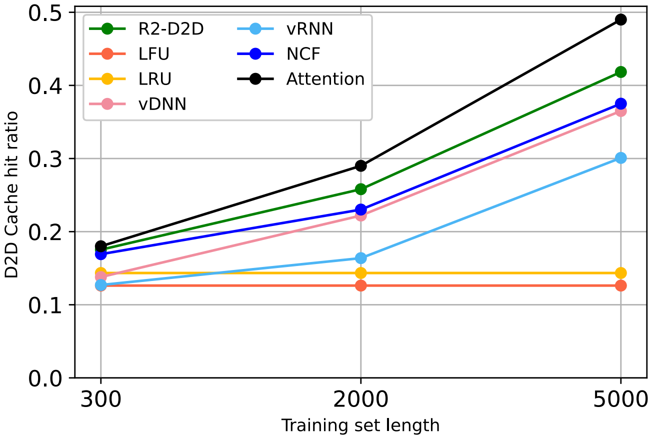

FIGURE 9. CHR_{D2D} - varying training set size.

as well. Further, the vRNN shows a steeper improvement in performance than the vDNN framework, which makes sense as the users are capable of caching more *popular* items, as predicted by the LSTMs.

As the number of UEs within a fixed-sized cell increases, both the number of neighbors for each user and the chances of cache hits increase. This is portrayed in Fig. 6 and Fig. 7, where the number of UEs is varied from 200 to 500 to 1000, and both traffic offloading and CHR_{D2D} increase with an increase in mobile users across all frameworks. For instance, when $M = 1000$, the CHR_{D2D} is approximately 0.45 for the R2-D2D and attention frameworks, whereas both the LRU and NCF underperform by almost 40% and 25%, respectively. Despite the minor improvement in the vDNN's traffic offloading, its CHR_{D2D} is almost equal, sometimes outperforming the vRNN, intimating a benefit to have a specific optimization goal for DNN for maximizing the CHR_{D2D}.

To test the real-world applicability of the proposed framework, we conduct several experiments by varying specific parameters. As shown in Figs. 8 and 9, D2D CHR is plotted with respect to the training dataset length. We have shown the CHR for 300, 2000, and 5000 data samples (training dataset length, or the number of rows) corresponding to the different content requests and cache hits. As anticipated, we found in Fig. 8 and Fig. 9 that vDNN and our stacked framework can best use the additional timesteps for CHR optimization. It is evident from Fig. 9, varying training set sizes from 300 to 5000, the vRNNs and vDNNs improvement in CHR_{D2D} was approximately 17% and 22%, respectively. Further, we see that the CHR_{D2D} sufficiently improves for the attention framework when training set size is increased, thereby establishing its superior performance capabilities.

V. CONCLUSION

In this paper, the DL based RNN and attention frameworks are proposed for caching in D2D networks. The proposed frameworks scale well with real-life scenarios, leaving scope for exploring any future developments in DL. Moreover, the mobility of users and the file popularity distribution are captured without using any unrealistic assumptions. In the first framework, we successfully used RNNs for modeling the stochastic variables of the environment and a special 1D-CNN for generating the CD. In the second framework, we successfully adapted the attention mechanism popularized in transformer architectures, used commonly in natural language processing, for the caching problem. We analyzed the performance of our frameworks in various scenarios and found that our proposed caching schemes perform better than the conventional frameworks such as NCF by 10% to 25%. We further conclude that the attention framework achieves superior performance across all settings.

REFERENCES

[1] Y. Huo, X. Dong, and W. Xu, "5G cellular user equipment: From theory to practical hardware design," *IEEE Access*, vol. 5, pp. 13992–14010, 2017.

[2] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the big data era," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 28–35, Jun. 2018.

[3] L. Li, G. Zhao, and R. S. Blum, "A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1710–1732, 3rd Quart., 2018.

[4] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[5] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142–149, Apr. 2013.

[6] G. H. S. Carvalho, I. Woungang, A. Anpalagan, M. Jaseemuddin, and E. Hossain, "Intercloud and HetNet for mobile cloud computing in 5G systems: Design issues, challenges, and optimization," *IEEE Netw.*, vol. 31, no. 3, pp. 80–89, May/Jun. 2017.

[7] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[8] B. Chen and C. Yang, "Caching policy for cache-enabled D2D communications by learning user preference," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6586–6601, Dec. 2018.

[9] C. Ssengonzi, O. P. Kogeda, and T. O. Olwal, "A survey of deep reinforcement learning application in 5G and beyond network slicing and virtualization," *Array*, vol. 14, Jul. 2022, Art. no. 100142.

[10] G. S. Paschos, A. Destounis, and G. Iosifidis, "Learning to cooperate in D2D caching networks," in *Proc. IEEE 20th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2019, pp. 1–5.

[11] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding performance of edge content caching for mobile video streaming," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1076–1089, May 2017.

[12] P. Gandotra, R. K. Jha, and S. Jain, "A survey on device-to-device (D2D) communication: Architecture and security issues," *J. Netw. Comput. Appl.*, vol. 78, pp. 9–29, Jan. 2017.

[13] M. Waqas *et al.*, "A comprehensive survey on mobility-aware D2D communications: Principles, practice and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1863–1886, 3rd Quart., 2020.

[14] K. Qi, S. Han, and C. Yang, "Learning a hybrid proactive and reactive caching policy in wireless edge under dynamic popularity," *IEEE Access*, vol. 7, pp. 120788–120801, 2019.

[15] S.-E. Elayoubi and J. Roberts, "Performance and cost effectiveness of caching in mobile access networks," in *Proc. 2nd ACM Conf. Inf.-Centric Netw.*, 2015, pp. 79–88.

[16] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[17] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, Oct. 2019.

[18] N. Zhao *et al.*, "Caching D2D connections in small-cell networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12326–12338, Dec. 2018.

[19] L. Ma, H. Zhang, T. Li, and D. Yuan, "Deep learning and social relationship based cooperative caching strategy for D2D communications," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2019, pp. 1–6.

[20] L. Li *et al.*, "Deep reinforcement learning approaches for content caching in cache-enabled D2D networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 544–557, Jan. 2020.

[21] S. Chakraborty, R. Bajpai, and N. Gupta, "R2-D2D: A novel deep learning based content-caching framework for D2D networks," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC-Spring)*, 2021, pp. 1–5.

[22] A. Vaswani *et al.*, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[23] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer Programming* (Graduate Texts in Mathematics). Cham, Switzerland: Springer Int., 2014.

[24] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

[25] W. Tang, G. Long, L. Liu, T. Zhou, J. Jiang, and M. Blumenstein, "Rethinking 1D-CNN for time series classification: A stronger baseline," 2020, *arXiv:2002.10061*.

[26] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th Int. Conf. Mach. Learn. Vol. 70*, 2017, pp. 933–941.

[27] S. Li *et al.*, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Advances in Neural Information Processing Systems*, vol. 32. Red Hook, NY, USA: Curran Assoc., Inc., 2019, pp. 5243–5253.

RAHUL BAJPAI received the M.Tech. degree in digital communication from ABV-IIITM, Gwalior, India, in 2014. He is currently pursuing the Ph.D. degree with the Department of Electrical and Electronics Engineering, BITS Pilani (K.K. Birla Goa Campus), India. His research interests are 6G technologies, non-orthogonal multiple access, full-duplex radio, millimeter-wave, device-to-device communications, intelligent reflecting surfaces, UAV-to-UAV communications, and machine learning for wireless communications. He has served as a Reviewer of the IEEE Access, *Wireless Networks* (Springer), and Wireless Personal communications (Springer).

SOURADEEP CHAKRABORTY received the B.E. degree in electrical and electronics engineering from BITS Pilani (K.K. Birla Goa Campus), Goa, India, in 2021. He is currently working as an Associate Product Manager with Flipkart. His interest areas include device-to-device communications, machine learning for wireless communications, and quantitative finance.

NAVEEN GUPTA received the M.Tech. degree in advanced communication systems from NIT Warangal in 2011, and the Ph.D. degree in wireless communications from IIIT Delhi, India in 2017. He is working as an Assistant Professor with the Department of EEE, BITS Pilani (K.K. Birla Goa Campus), India. His research interests are resource allocations for next generation wireless communication techniques, nonorthogonal multiple access, full-duplex, intelligent reflecting surfaces, UAV-to-UAV communications, millimeter-wave, and device-to-device communication. He has served as a Reviewer of the IEEE Transactions on Cognitive Communications and Networking, IEEE Transactions on Communications, IEEE Access, and the IEEE Transactions on Wireless Communications.