IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON WIRELESS CACHIING TECHNIQUE FOR 5G

Received February 7, 2017, accepted February 23, 2017, date of publication March 7, 2017, date of current version May 17, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2678990

Adaptive Scheme for Caching YouTube Content in a Cellular Network: Machine Learning Approach

S. M. SHAHREAR TANZIL[1], WILLIAM HOILES[1], AND VIKRAM KRISHNAMURTHY[2], (Fellow, IEEE)
[1]Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada
[2]Department of Electrical and Computer Engineering, Cornell Tech and Cornell University, New York, NY 10011, USA

Corresponding author: S. M. S. Tanzil (stanzil@ece.ubc.ca)

ABSTRACT Content caching at base stations is a promising solution to address the large demands for mobile data services over cellular networks. Content caching is a challenging problem as it requires predicting the future popularity of the content and the operating characteristics of the cellular networks. In this paper, we focus on constructing an algorithm that improves the users' quality of experience (QoE) and reduces network traffic. The algorithm accounts for users' behavior and properties of the cellular network (e.g. cache size, bandwidth, and load). The constructed content and network aware adaptive caching scheme uses an extreme-learning machine neural network to estimate the popularity of content, and mixed-integer linear programming to compute where to place the content and select the physical cache sizes in the network. The proposed caching scheme simultaneously performs efficient cache deployment and content caching. Additionally, a simultaneous perturbation stochastic approximation method is developed to reduce the number of neurons in the extreme-learning machine method while ensuring a sufficient predictive performance is maintained. Using real-world data from YouTube and a NS-3 simulator, we demonstrate how the caching scheme improves the QoE of users and network performance compared with industry standard caching schemes.

INDEX TERMS Content caching, cellular network, extreme learning machines, mixed-integer linear programming, popularity prediction, feature selection, YouTube.

I. INTRODUCTION

Cellular networks are experiencing substantial growth in data traffic as a consequence of increasing demand for rich multimedia content via mobile devices. However, it is challenging for cellular network operators to maintain users' QoE while keeping up with this massive growth in mobile data traffic. The expected data traffic served by cellular networks is expected to reach 30.6 exabytes (10^{18}) per month by the year 2020, an eightfold increase compared to 2015 [1]. Content caching in cellular networks is a potential solution to address the large demands for mobile data services over cellular networks [2]–[4]. Examples include caching at eNodeB [5] and caching at home eNodeB [6]. Caching in cellular networks is an attractive solution as data storage is inexpensive compared with increasing the bandwidth of a cellular network. It has been illustrated that almost 60% of the mobile data traffic results from video traffic [1]. Additionally, the increased traffic typically results for multiple requests for a few highly popular video content [7]. By only caching the highly popular content at the base stations (BS), user demand for the same content can be served locally. This reduces the overall network traffic and improves the QoE for users requesting content.

The literature on content caching in cellular networks has grown in recent years [8]–[10]. Caching methods roughly fall into two categories: *i)* designing new content caching methods with different objectives e.g., minimizing downloading delay, energy consumption, network congestions or maximizing users' QoE while assuming content popularity is known; and *ii)* developing new methods for predicting content popularity and caching the most popular content. For instance, [6] presents content caching methods for BSs to minimize content downloading delay. The proposed coded content caching optimization problem in [6] is convex and involves the solution to a linear program. In [9], a multicast-aware caching method is developed that minimizes energy consumption in the network while the method described in [5] improves users' QoE. A cooperative content caching policy is proposed in [7] to improve network performance and users' QoE. The presented content caching problem is

formulated as an integer linear programming and suboptimal solutions are devised using the hierarchical primal-dual decomposition method.

A limitation with the caching methods [6], [7], [9] is that they assume knowledge of the content popularity in advance, and assume that the content popularity follows a Zipf distribution. In reality, content popularity must be estimated [11], [12]. In [13]–[15], content popularity is estimated using the request statistics of the content.[1] The content is then cached based on the estimated content popularity. In [16] and [17], collaborative filtering methods are used to cluster users with similar content preferences, and then cache the content based on the number of users in each cluster. For new users, their social network characteristics can be used to estimate their content preferences and cluster association. The combination of collaborative filtering with social network information is used in [18] and [19] to mitigate the *cold start* and *data sparseness* issues associated with collaborative filtering. There are two main issues with employing these methods for caching content. The first is that these methods are highly intrusive as the methods require knowing the user's content requests and social network information–for example, these methods can not be employed in countries such as Canada without user consent as it would violate privacy laws.[2] The second issue is that content popularity is estimated using the content request statistics. Therefore, these methods can not be used to cache new content which have no user request statistics.

In this paper, we construct an adaptive caching scheme that accounts for users' behavior and properties of the cellular network (e.g. cache size, bandwidth, and load). The scheme does not require specific user's content requests and/or social network information. Instead, the popularity of the content is predicted using features of the content. This allows the caching of popular content without the need for directly observing user requests for the content. Additionally, since the popularity of the content is predicted, the content caching can be performed when the network load is minimal reducing in the overall energy usage of the network. A schematic of the adaptive caching scheme proposed in this paper is displayed in Fig. 1. The main contributions of this paper include:

- A machine learning algorithm, based on the extreme learning machine (ELM) [20], to estimate the popularity of content based on the features of the content. A simultaneous perturbation stochastic approximation algorithm is also constructed to perform feature-selection and design parameter selection to improve the performance of the machine learning algorithm.
- The features in the ELM are constructed using a combination of human perception models and network parameters. For YouTube content, which contains text and images, the human perception model is used to compute the brightness and contrast as perceived by a human. Additionally for text, the sentiment and subjectivity[3] of the text are computed using a human perception model.
- A mixed-integer linear program (MILP) is constructed to perform cache initialization that accounts for the predicted content popularity and properties of the cellular network.
- An adaptive caching scheme which uses the MILP for cache initialization, and the S3LRU (Segmented Least Recently Used with three segments) cache replacement scheme for dynamically adjusting the cache based on the requests from users. The combination of these schemes increases the overall performance of the cellular network.
- The performance of the adaptive caching scheme is illustrated using real-world data from YouTube and a NS-3 simulator. The results illustrated that the adaptive caching scheme improves network performance and users' QoE compared with industry standard caching schemes [6], [17], [21].

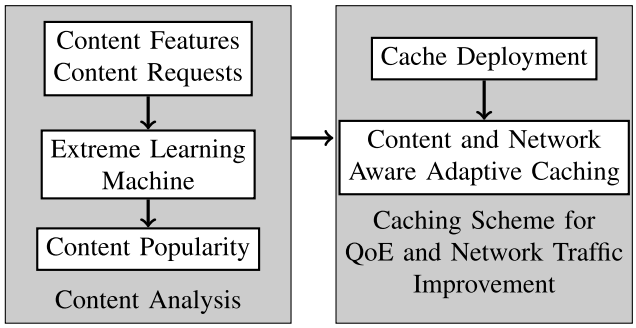

FIGURE 1. Schematic of the caching scheme. Improving the quality of experience (QoE) involves ensuring a higher cache hit ratio for requested content and reduced downloading delay.

The paper is organized as follows. The system model and problem formulation are presented in Sec.II. The content and network aware adaptive caching scheme, which accounts for the parameters of the content popularity and technological network, is presented in Sec.III. In Sec.IV we describe how ELMs can be used to efficiently estimate content popularity using both content features and the request statistics of users as they become available. The performance of ELM for caching, and content and network aware adaptive caching scheme are illustrated in Sec.V using real-world data from YouTube.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a heterogeneous cellular network in a geographical region where base stations (BSs), such as eNodeB and home eNodeB, are deployed and equipped with a physical storage/cache capacity. The network shown in Fig. 2 can be represented by a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of vertices $\mathcal{V}$ is used to denote the set of cache enabled BSs which comprises

[1] In this paper content refers to YouTube video files.

[2] https://www.loc.gov/law/help/online-privacy-law/canada.php

[3] Sentiment is a measure of how humans feel about a piece of writing typically measured on the scale of positive, neutral, or negative. Subjectivity is related to identifying if the written text is factual (subjective) or an expression of an opinion (objective).

TABLE 1. Glossary of parameters.

Parameters	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Network graph
$\mathcal{V}$	Set of cache-enabled base stations
$V = \lvert\mathcal{V}\rvert$	Number of cache-enabled base stations
$\mathcal{E}$	Set of communication links
$\mathcal{J}$	Set of content
$J = \lvert\mathcal{J}\rvert$	Number of content
$\mathcal{C}$	Set of categories
$C = \lvert\mathcal{C}\rvert$	Number of categories
f_j	Size of content $j \in \mathcal{J}$
Popularity Estimation	
T	Total number of observations
t	Time index
$\mathcal{D} = \{x_j, v_j(t)\}$	Observation dataset
$h_k(x; \theta_k)$	Transfer function for hidden-layer neuron k
θ_k	Parameters of hidden-layer neuron k
β_k	Output weights
L	Total neurons of ELM
l_{th}	Video popularity threshold

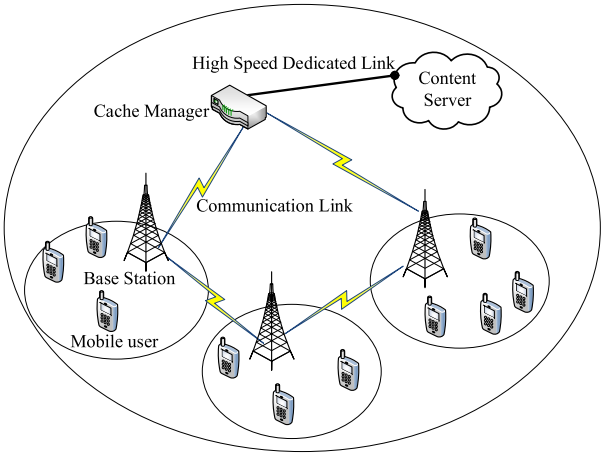

FIGURE 2. A typical network architecture. Base stations (BSs) are connected with each other and with the cache manager (CM) via heterogeneous communication links. Cache manager is connected to the content server via high speed dedicated link.

of V BSs and indexed by $i \in \mathcal{V} = \{1, 2, \ldots, V\}$. The set of edges $\mathcal{E}$ denotes communication links among BSs. BSs can communicate with each other and with a cache manager (CM) via Xn interface [22], [23]. CM is connected to a content server such as a telco content delivery network (CDN) via a high-speed dedicated link and is responsible for:

i) retrieving unavailable content from the content server;

ii) maintaining a lookup table that stores cached content location in the network;

iii) forwarding content request to the neighbouring BS which has the content;

iv) gathering information from BSs about the content are being requested;

v) making decision when to refresh entire cache of the BSs which can be done either specific intervals or when content popularity changes significantly;

vi) performing computations for adaptive caching.

Mobile users are connected to the BSs according to a cellular network protocol. Connected BS is responsible for serving users' content requests. If a requested content is in the cache of the connected BS, the request is served instantly. In this case, the content downloading delay is lower, and hence, improves user's QoE. In addition, no additional load is put on the back-haul connection which reduces network traffic. On the other hand, when a requested content is not available at the connected BS, the request is forwarded to the CM. The CM checks the lookup table whether the requested content is available in the network. If the content is available in the network, CM performs all the required signaling to fetch the content from the neighbour BS. Content served by the neighbour BSs incur lower downloading delay and reduce network traffic. Finally, CM fetches content from the content server when requested content is unavailable in the network or when retrieving content from neighbour BSs incurs higher delay than the content server.

The content that can be cached is indexed by $\mathcal{J} = \{1, 2, \ldots, J\}$. Let f_j denote the size of the j-th content. The initial file transferring cost via the CM to a BS i is denoted by d^{gi} (second per byte), and the latency between BS i and BS l is denoted by d^{il} where $l \in \mathcal{V}$. d^{gi} and d^{il} both depends on the network topology, communication link, and routing strategy. The network topology may vary over time and routing strategy can be adjusted according to the network traffic. d^{gi} and d^{il} also depends on channel quality when BSs are communicating with one another via a wireless link. A glossary of the parameters used throughout the paper is provided in Table 1.

III. CONTENT AND NETWORK AWARE ADAPTIVE CACHING SCHEME FOR CELLULAR BASE STATIONS

Our proposed content and network aware adaptive caching scheme proceeds as follows. Given the estimated content popularity, network topology, link capacity, routing strategy and cache deployment budget/energy usage budget in the network, the adaptive caching scheme prescribes individual BSs which content to cache and adapts its prescriptions as the preferences of users (content popularity) evolves over time. The caching scheme utilizes popularity estimation to account for the users content request characteristics. The benefit of using popularity estimators in the caching decision is that it allows caching decisions to be made–that is when the network is not being heavily utilized, popular content can be transferred between BSs without hindering the quality of service of the network.

The rest of this section is organized as follows: Sec.III-A formulates the caching scheme as a mixed-integer linear programming (MILP) while Sec.III-B provides implementation considerations of the proposed caching scheme.

A. MIXED-INTEGER LINEAR PROGRAM FORMULATION

The adaptive caching scheme takes into account content popularity, link capacity, network topology, cache size, and network operating costs. The network operating costs include storage read/write costs and the cost of data transmission in the network. In this paper, energy usage to read/write files from hardware units are considered as cache deployment cost.

Hardware units draw energy when they are active due to read/write of the cached content. On the other hand, hardware units do not cache content when they are in sleep/idle mode and draw a negligible amount of energy. Therefore, higher active hardware units mean higher storage/cache size for caching at a higher energy cost to operate. Network operators allow BSs to activate a certain number of hardware unit(s) for caching. The flexible cache size facilitates network operators to provide physical storage at different BSs according to their content popularity distribution and network parameters such as link capacity and network topology while maintaining a target cache deployment cost in the network at a given time.

In the network, each BS can select a maximum of R possible hardware units (e.g. physical cache storage sizes) where each hardware unit has a storage size of s_0. Each BS can only use $r^i \in \{1, \ldots, R\}$ active hardware unit(s) due to the cache deployment cost constraint. Each hardware unit that is activated has an associated cost defined by z_0. The maximum physical storage size that can be used in the network at any given time to maintain target cache deployment cost is denoted by S. The parameter $\hat{\mu}_j^i(t) \in [0, 1]$ represents the estimated popularity of content j at BS $i \in \mathcal{V}$ for time index $t \in \{1, \cdots, T\}$. The parameter $\hat{\mu}_j^i(t)$ is computed by:

$$\hat{\mu}_j^i(t) = \frac{\hat{v}_j^i(t)}{\sum_{j \in \mathcal{J}} \hat{v}_j^i(t)}, \tag{1}$$

where $\hat{v}_j^i(t)$ is total views of content j at BS i for time index t. In Sec.IV we provide a method to estimate the popularity of content based on the features of the content.

The MILP is formulated in (2) which minimizes the content downloading delay, taking into account initial file transferring cost, and cache deployment cost in the network while maintaining total cache deployment cost. There are three decision variables in the MILP:
i) $r^i \in \{1, \ldots, R\}$ denotes the number of memory units used at BS i. The total size of the physical cache used at BS i is equal to $r^i s_0$ where s_0 is the physical size of the memory units (e.g. one hardware unit may represent 200 GB of physical memory);
ii) $a_j^i \in \{0, 1\}$ which is equal to 1 if content j is cached by BS i;
iii) b_j^{il} which represents the fraction of content j served by BS i to BS l. Note that $b_j^{ll} = 1$ means that BS l caches the content j and serves the request itself. Note, the time index t is omitted for brevity in the content popularity and decision variables.

$$\min_{b_j^{il}, a_j^i, r^i} \Big(w_1 \sum_{j \in \mathcal{J}} \sum_{\substack{i \in \mathcal{V} \\ l \in \mathcal{V}}} f_j \hat{\mu}_j^l d^{il} b_j^{il} + w_2 \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{V}} f_j d^{gi} a_j^i + w_3 \sum_{i \in \mathcal{V}} r^i z_0 \Big) \tag{2}$$

subject to constraints

$$\sum_{i \in \mathcal{V}} b_j^{il} = 1 \quad \forall j \in \mathcal{J},\ l \in \mathcal{V} \tag{3}$$

$$b_j^{il} \leq a_j^i \quad \forall j \in \mathcal{J}, i \in \mathcal{V},\ l \in \mathcal{V} \tag{4}$$

$$\sum_{j \in \mathcal{J}} f_j a_j^i \leq r^i s_0 \quad \forall i \in \mathcal{V} \tag{5}$$

$$\sum_{i \in \mathcal{V}} r^i s_0 \leq S \tag{6}$$

$$b_j^{il} \geq 0 \quad \forall j \in \mathcal{J}, i \in \mathcal{V},\ l \in \mathcal{V} \tag{7}$$

$$a_j^i \in \{0, 1\} \quad \forall j \in \mathcal{J}, i \in \mathcal{V} \tag{8}$$

$$r^i \in \{1, 2, \cdots, R\} \quad \forall i \in \mathcal{V}. \tag{9}$$

The first term of the objective function in the MILP (2) accounts for the content downloading delay in the network. The second term of equation (2) represents the initial content transferring cost in the network. The third term reflects cache deployment cost in the network. w_1 and w_2 are the weight of real-time latency/downloading delay cost and initial file transferring cost in the objective function, respectively. w_3 reflects the weight of cache deployment cost in the objective function. Constraint (3) ensures that total fraction of j-th content is equal to 1. Constraint (4) represents the fact that BS i can serve other BSs' request only when it caches the requested content. Constraint (5) ensures that each BS i fully uses the available cache where f_j is the size of the j-th content, s_0 is the size per unit of physical storage, and r^i is the number of units of storage. Constraint (6) maintains the cache deployment budget in the network.

B. IMPLEMENTATION CONSIDERATIONS

1) MILP SOLUTION

The MILP (2) is NP-hard [24], [25]. Due to the size of the problem such as the number of available content and the number of network nodes, it is intractable to find optimal solutions in real-time. However, several numerical methods exist for estimating the solution to (2) which include: Branch-and-bound, cutting planes, branch-and-cut, branch-and-price are popular heuristic approaches to solve MILP via linear relaxation [13], [26]–[29]. In this paper, individual videos are grouped into clusters $c \in \mathcal{C}$ where $\mathcal{C} = \{1, \cdots, C\}$ represents the set of cluster/category of videos. Machine learning methods can be used to estimate the optimal clusters, however, in the YouTube network a suitable clustering method is to cluster the YouTube videos based on their associated category. Examples of categories of YouTube videos include "Entertainment", "Music", "News", "Sports", "Howto". The set of content in category $c \in \mathcal{C}$ is denoted by $J^c \subseteq \mathcal{J}$. The popularity associated with each category $c \in \mathcal{C}$ at base station $i \in \mathcal{V}$ is given by:

$$\hat{\mu}^{ic}(t) = \sum_{j \in J^c} \hat{\mu}_j^i(t) \tag{10}$$

where $\hat{\mu}_j^i(t)$ is computed using (1). Given the categorical content popularity (10), the MILP can be used to optimally select where to cache these content.

2) CACHE UPDATE FREQUENCY

In the adaptive caching scheme there are two content caching schemes used. The first is the MILP which performs the

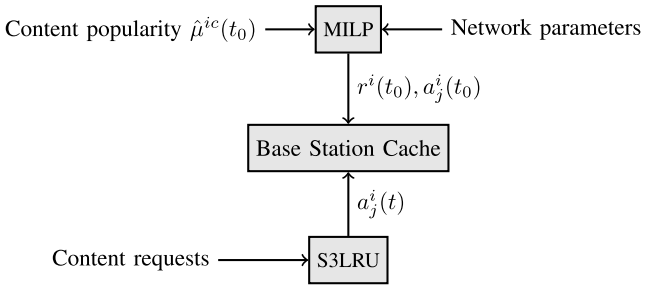

FIGURE 3. A schematic of the adaptive caching scheme. Initially the MILP uses the estimated content popularity $\hat{\mu}^{ic}(t_0)$ (10) and network parameters to compute the physical cache size $r^i(t_0)s_0$ to be used at base station $i \in \mathcal{V}$, and $a^i_j(t_0) \in \{0, 1\}$ indicating if content $j \in \mathcal{J}$ is to be cached at base station $i \in \mathcal{V}$ initially. t_0 indicates when the solution of the MILP is used to update the base station cache. Then, the S3LRU uses the content requests from users to compute $a^i_j(t) \in \{0, 1\}$ at times $t > t_0$.

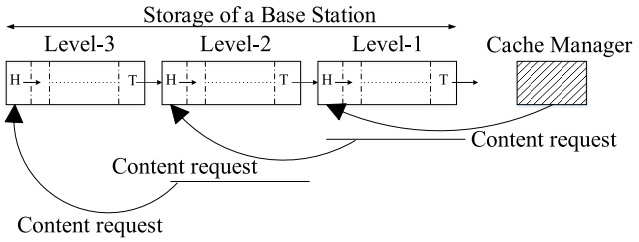

FIGURE 4. A schematic of the segmented least recently used (S3LRU) cache replacement scheme with three levels denoted by Level-3, Level-2, and Level-1. The S3LRU is used to control the cache content at each base station after the cache is initialized. Level-3 has the content with the highest popularity, and Level-1 has the content with the lowest popularity. In each Level, the most popular content is placed in the head *H* of the level, and the least popular content is placed in the tail *T* of the level. If the requested content is not currently in the cache, then the requested content is transferred from the cache manager (CM) to the head segment of the Level-1 cache. If the requested content is currently in the cache, then the content is moved to the head of the next level with all other content shifted down by one with the content in the tail of Level-1 removed from the cache.

cache initialization, and the second caching scheme which uses users' request statistics to dynamically cache content. An important design parameter to consider when using the MILP (2) for cache initialization is when to replace the currently cached content. Given that the MILP may replace a significant portion of the cached content, typically the solution of the MILP will only be used when the network traffic flow is minimal. Fig. 3 provides a schematic of the adaptive caching scheme.

The BSs initialize their physical cache size and content to cache based on the results of the MILP. Then, the S3LRU [30] is used to select the content to cache based on the users' requests. In the S3LRU caching scheme, the physical cache of each BS is composed of three segments as illustrated in Fig. 4. The segments are defined by Level-3, Level-2, and Level-1 in which Level-3 has the content with the largest popularity, and Level-1 has the content with the lowest popularity. Each of the Levels is composed of a head segment and a tail segment where the head segment contains the most popular content, and the tail segment contains the least popular content in the associated level. The dynamics how the content is cached in the S3LRU is illustrated in Fig. 4. If the requested content is currently in the cache, then the content is moved to the head of the next level with all other content shifted down by one. Note that in Level-1, the content in the tail segment is evicted from the cache (that is, it no longer resides in the cached content). If the requested content is not currently in the cache, then the requested content from the cache manager is placed in the head segment of Level-1 of the cache. This replacement policy ensures that the popular content resides in Level-3 of the cache, and the least requested content resides in Level-1 of the cache.

IV. EXTREME LEARNING MACHINE (ELM) FOR POPULARITY PREDICTION

The adaptive caching scheme in Sec.III requires the future popularity of the content to be known. In this section we use ELMs [31], [32] to estimate the popularity of content given the content features and previous request statistics of the content. Additionally, we provide methods to optimize the number of neurons of the extreme learning machine and select the optimal features to perform the popularity prediction. As an illustrative example, we focus on predicting the popularity of videos in YouTube. The prediction of popular content in YouTube is challenging as the features of YouTube video contains significant noise. Therefore the machine learning algorithms used must be able to address this challenging problem of mapping from these type of noisy features to the associated popularity of a video. Of the machine learning methods tested we found that the ELM [31], [32] provides sufficient performance to estimate the popularity of YouTube videos. Though the results presented in this section are focused on the use of ELM, the constructed features, neuron selection algorithm, and feature selection algorithm are general and can be used with other machine learning techniques.

A. PREDICTING CONTENT POPULARITY WITH EXTREME LEARNING MACHINES

Consider a dataset $\mathcal{D} = \left\{\{x_j, v_j(t)\} : j \in \mathcal{J} = \{1, \ldots, J\}, t \in \{1, \ldots, T\}\right\}$ of features $x \in \mathbb{R}^M$, and total views $v_j(t)$ on day t for content $j \in \mathcal{J}$. The aim is to construct a model that relates the features x to the total views v based on the dataset $\mathcal{D}$. For example, single hidden-layer feed-forward neural networks can be used for estimating the functional relationship $v_j(t)$ and the features x_j. However, in practice the selection of the model and training method is complex requiring consideration of the universal approximation ability of the model, sequential learning ability, efficiency, parallel implementation, and hardware implementation. Recently, based on the Rosenblatt perceptron [33], ELMs [20] have been introduced for learning the functional relationship between inputs x_j and output $v_j(t)$. The ELM which satisfies the universal approximation condition [34], [35], can be implemented in parallel [31], can be trained sequentially for large datasets or as new training data becomes available [36], [37], and can be efficiently implemented on field-programmable gate array devices as well as

complex programmable logic devices [32]. The ELM is a single hidden-layer feed-forward neural network in which the parameters of the hidden-layer are randomly generated by a distribution, and the subsequent output weights are computed by minimizing the error between the computed output $v_j(t)$ and the measured output from the dataset $\mathcal{D}$. Each hidden-layer neuron can have a unique transfer function. Popular transfer functions include the sigmoid, hyperbolic tangent, and Gaussian however any non-linear piecewise continuous function can be utilized.

The classic extreme learning machine, presented in [38], is given by:

$$\hat{v}_j(t) = \sum_{k=1}^{L} \beta_k h_k(x_j; \theta_k) \tag{11}$$

with $\beta_1, \beta_2, \ldots, \beta_L$ the weights of each neuron, $h_1(x_j), h_2(x_j), \ldots, h_L(x_j)$ the associated transfer function of each neuron, and $\hat{v}_j(t)$ the estimated total views of the video content j at time t. Given $\mathcal{D}$, how can the ELM model parameters β_k, θ_k, and L in (11) be selected? For fixed number of hidden neurons L, the ELM trains β_k and θ_k in two steps. First, the hidden layer parameters θ_k are randomly initialized. Any continuous probability distribution can be used to initialize the parameters θ_k. Second, the parameters β_k are selected to minimize the square error between the model output and the measured output from $\mathcal{D}$. Formally,

$$\beta^* \in \underset{\beta \in \mathbb{R}^L}{\operatorname{argmax}} \left\{ ||H\beta - Y||_2^2 \right\} \tag{12}$$

with H the hidden-layer output matrix with entries $H_{kj} = h_k(x_j; \theta_k)$ for $k \in \{1, 2, \ldots, L\}$ and $j \in \mathcal{J}$, and Y the target output with entries $Y = [y_1, y_2, \ldots, y_J]$. The solution of (12) is given by $\beta^* = H^+ Y$ where H^+ denotes the Moore-Penrose generalized inverse of H. Several efficient methods can be used to compute β^* (refer to Golub and Van Loan, 2012). The benefit of using the ELM, (11) and (12), is that the training only requires randomly generating parameters θ_k; the parameters β_k are computed as the solution of a linear algebraic system of equations.

B. FEATURE CONSTRUCTION FOR POPULARITY PREDICTION

Here we describe how the features of YouTube videos are constructed using the YouTube Application Programming Interface.[4]

The meta-data of each YouTube video contains four primary components: Thumbnail, Title, Keywords (also known as tags), and Description. Additionally, each YouTube video is associated with a Channel that contains features such as the number of subscribers. The viewcount of a video is sensitive to the features of the Thumbnail, Title, Keywords, and Channel. However, features associated with the description appear not to significantly impact the viewcount of a video. This may result because when performing video searched on YouTube, only a subset of the description is provided to the users. In this paper we focus on how features of the Thumbnail, Title, Keywords, and Channel can be used to estimate the the viewcount of a YouTube video. Note that our analysis does not include the video or audio quality, and the description of the YouTube video. These features will impact the dynamics of users subscribing to a channel, and rating the video, however, they do not directly impact the viewcount of a specific video.

For the Thumbnail, 19 features are computed which include: the blurriness (CannyEdge, Laplace Frequency), brightness, contrast, overexposure, and entropy of the thumbnail. All image analysis is performed using the OpenCV (Open Source Computer Vision) library.[5] To compute the brightness η_w of each video thumbnail, we first import the thumbnail in RGB color format. Let us denote X_{ue} as a pixel in the thumbnail image, and $R(X_{ue}) \in [0, 255]$ as the red light, $G(X_{ue}) \in [0, 255]$ as the green light, and $B(X_{ue}) \in [0, 255]$ as the blue light associated with pixel X_{ue}. The total size of the thumbnail is given by $N_X N_Y$ with $u \in \{1, \ldots, N_X\}$ and $e \in \{1, \ldots, N_Y\}$. The brightness of the image is then computed using:

$$\begin{aligned} \eta_w(X_{ue}) &= 0.299R(X_{ue}) + 0.587G(X_{ue}) + 0.114B(X_{ue}) \\ \eta_w &= \frac{1}{765 N_X N_Y} \sum_{u=1}^{N_X} \sum_{e=1}^{N_Y} \eta_w(X_{ue}). \end{aligned} \tag{13}$$

Typically humans' perceived brightness for color are most sensitive to variations in green light, less to red, and least to blue. The coefficients in (13) are associated with the perceived brightness for color, and the specific values are obtained from the OpenCV software. The contrast of each thumbnail ζ_w is computed using the RMS Contrast given by:

$$\zeta_w = \sqrt{\frac{1}{765 N_X N_Y} \sum_{u=1}^{N_X} \sum_{e=1}^{N_Y} \left(\eta_w(X_{ue}) - \eta_w\right)^2}. \tag{14}$$

As we illustrate, the brightness η_w and contrast ζ_w of a videos Thumbnail provide important information that can be used to estimate the viewcount of a YouTube video.

For the Title, 23 features are computed which include: word count, punctuation count, character count, Google hits (e.g. if the title is entered into the Google search engine, how many results are found), and the Sentiment/Subjectivity of the title computed using Vader [39], and TextBlob.[6] For the Keywords, 7 features are computed which include: the number of keywords, and keyword length. In addition, to the above 49 features, we also include auxiliary video and channel features including: the number of subscribers, resolution of the thumbnail used, category of the video, the length of the video, and the first-day viewcount.

In total 54 features are computed for each video. The complete dataset used for the sensitivity analysis is given by

[4] Specific details on how to interact with the YouTube API are provided at https://developers.google.com/youtube/v3/docs/.

[5] http://opencv.org/

[6] http://textblob.readthedocs.io/en/dev/

$\mathcal{D} = \{(x_j, v_j)\}_{j\in\mathcal{J}}$, with $x_j \in \mathbb{R}^{54}$ the computed features for video $j \in \mathcal{J}$, v_j the viewcount $t = 14$ days after the video is published, and J the total number of videos used for the sensitivity analysis.

C. OPTIMIZING THE NUMBER OF NEURONS IN THE EXTREME LEARNING MACHINE

For online applications of caching where millions of videos may be cached, it is critical to consider the computational cost of evaluating the popularity of the content. In this section we consider how to select the number of neurons L in the ELM while still ensuring a sufficient predictive performance is maintained.

Several methods exist for selecting the number of neurons L (11) in the extreme learning machine [34], [40], [41]. In [41] a multiresponse sparse regression is utilized to order the neurons from best to worst. Then using the leave-one-out generalization metric the optimal number of neurons can be selected. Another method is to incrementally increase L until the desired accuracy or maximum number of neurons is reached [34]. In [40] neurons are added incrementally until the output of the ELM negligibly effected as measured using a non-parametric noise estimator known as the delta test. The main idea in [40] is that if the increase in accuracy of the ELM is above the estimated variance of the ELM then a neuron is added.

The predictive performance (e.g. probability of Type-I and Type-II errors) of the ELM, as a function of the number of neurons L, is a random variable as a result of how each ELM is initialized. Given the desired predictive performance, instead of having to estimate the mean of the ELM for each L and then using a gradient decent method, one could instead employ the stochastic perturbation simultaneous approximation (SPSA) [42] method to compute the optimal number of neurons. The ELM parameter L is adapted to estimate:

$$\underset{L\in\{1,2,\dots\}}{\arg\min}\ A(L) = \mathbb{E}\left[\mathbb{P}(\text{Type-I error}) + \mathbb{P}(\text{Type-II error}) + g\tau\right] \tag{15}$$

where τ is the training time of the ELM, and g is a design parameter. Here $\mathbb{E}$ denotes the expectation with respect to the random variable θ defined in (11), and $\mathbb{P}$ denotes the probability. Since the probability of Type-I errors, Type-II errors, and the training time τ is not known explicitly, (15) is a simulation based stochastic optimization problem. To determine a local minimum value of $A(L)$, several types of stochastic optimization algorithms can be used [42]. In this paper we use the following SPSA algorithm (Algorithm 1):

The SPSA is a gradient based stochastic optimization algorithm where the gradient is estimated numerically by random perturbation(17). The nice property of the SPSA algorithm is that estimating the gradient $\nabla_L A_n(L_n)$ in (17) requires only two measurements of the cost function (16) corrupted by noise per iteration. See [42] for a tutorial exposition of the SPSA algorithm. For decreasing step size $\psi = 1/n$, the SPSA algorithm converges with probability one to a local stationary point. For constant step size ψ, it converges weakly (in probability) to a local stationary point.

Algorithm 1 SPSA Neuron Selection

Step 1: Choose initial ELM parameters L_0 by generating each from the distribution $\mathcal{N}(0, 1)$, and define the video popularity threshold as l_{th}.

Step 2: For iterations $n = 1, 2, 3, \dots$

- Estimate the Type-I and Type-II error probabilities of the ELM with L_n neurons using

$$\begin{aligned}
\text{FP} &= \sum_{j=1}^{|\mathcal{J}|} \mathbf{1}\{(\hat{v}_j \geq l_{\text{th}}) \cap (v_j < l_{\text{th}})\},\\
\text{TP} &= \sum_{j=1}^{|\mathcal{J}|} \mathbf{1}\{(\hat{v}_j \geq l_{\text{th}}) \cap (v_j \geq l_{\text{th}})\},\\
\text{FN} &= \sum_{j=1}^{|\mathcal{J}|} \mathbf{1}\{(\hat{v}_j < l_{\text{th}}) \cap (v_j \geq l_{\text{th}})\},\\
\text{TN} &= \sum_{j=1}^{|\mathcal{J}|} \mathbf{1}\{(\hat{v}_j < l_{\text{th}}) \cap (v_j < l_{\text{th}})\},\\
\mathbb{P}(\text{Type-I error}) &\approx FP/(TP + FN),\\
\mathbb{P}(\text{Type-II error}) &\approx FN/(TN + FP),
\end{aligned} \tag{16}$$

  where $\mathbf{1}\{\cdot\}$ is the indicator function and $\cap$ denotes the logical and operator. Given (16), compute the cost $\hat{A}_n(L_n)$ by substituting (16) into (15).
- Compute the gradient estimate $\hat{\nabla}_L \hat{A}_n(L_n)$:

$$\hat{\nabla}_L \hat{A}_n(L_n) = \frac{\hat{A}_n(L_n + \Delta_n\omega) - \hat{A}_n(L_n - \Delta_n\omega)}{2\omega\Delta_n}$$
$$\Delta_n(j) = \begin{cases} -1 & \text{with probability } 0.5 \\ +1 & \text{with probability } 0.5 \end{cases} \tag{17}$$

  with gradient step size $\omega > 0$.
- Update the number of neurons L_n of the ELM at step n with step size $\psi > 0$:

$$L_{n+1} = L_n - \psi\hat{\nabla}_L \hat{A}_n(L_n).$$

D. STOCHASTIC FEATURE SELECTION

Feature selection algorithms are geared towards selecting the minimum number of features such that a sufficiently accurate prediction is possible. If the feature set is too large then the generalization error of the predictor will be large. Though several feature selection algorithms exist [43], [44], only the ELM feature selection method presented in [45] has utilized feature selection to improve the performance of the ELM. In this section we construct a feature selection algorithm, Algorithm 2, which relies on computing the optimal features based on the model sensitivity to variations in the features and an estimate of the generalization error of the model. Features are removed sequentially while ensuring the generalization error is sufficiently low.

The main idea of the sequential feature selection algorithm (Algorithm 2) is to sequentially remove the least useful features while ensuring that the performance of the ELM is sufficiently high. This is performed by computing the output of the ELM with all features, then computing the output with one of the features held constant at its mean (i.e. the null ELM model). If the output from the ELM and null ELM are similar under some metric then the feature held constant does not contribute significantly to the predictive performance of the ELM and should be removed. This process is repeated sequentially in Algorithm 2 until a performance threshold is reached.

Algorithm 2 Sequential Wrapper Feature Selection

Step 0: Collect the dataset $\mathcal{D} = \left\{\{x_j, v_j\} : j \in \mathcal{J} = \{1, \ldots, J\}\right\}$ of features $x_j \in \mathbb{R}^M$ and video view count v_j for videos. Select the desired similarity metric $F(\cdot)$ (e.g. R^2 coefficient of determination).

Step 1: Train the ELM (11) using the dataset $\mathcal{D}$ and (12). Denote the predicted viewcount from the ELM by $\hat{v}_{\mathcal{D}}$.

Step 2: For $m \in \{1, 2, \ldots, M\}$, train the ELM using the dataset $\mathcal{D}^m$ where $\mathcal{D}^m$ is the dataset $\mathcal{D}$ with the feature $x_j(m)$ held at its mean for all $j \in \mathcal{J}$. Denote the predicted output from each of the $m \in \{1, 2, \ldots, M\}$ ELMs by $\hat{v}^m_{\mathcal{D}}$.

Step 3: Compute the feature index m with maximum similarity between $\hat{v}_{\mathcal{D}}$ from Step 1 and $\hat{v}^m_{\mathcal{D}}$ from Step 2:

$$m^* \in \underset{m \in \{1,\ldots,M\}}{\operatorname{argmax}} \{F(\hat{v}_{\mathcal{D}}, \hat{v}^m_{\mathcal{D}})\} \tag{18}$$

where $F(\cdot)$ denotes the selected similarity metric from Step 0.

Step 4: Compute the metrics of performance (Type-I and Type-II error probabilities) using the ELM trained using the dataset $\mathcal{D}^*$ where the feature m^* from Step 3 has been removed. If the metrics of performance are too high then stop. Otherwise return to Step 1 using the dataset $\mathcal{D} \leftarrow \mathcal{D}^*$.

V. NUMERICAL EXAMPLE OF CONTENT AND NETWORK AWARE ADAPTIVE CACHING USING REAL-WORLD YouTube DATA

This section provides a numerical example to illustrate the performance of the adaptive caching using real-world YouTube dataset. Sec.V-A describes simulation setup. Performance of extreme learning machine for caching is presented in Sec.V-B. We use results obtained from Sec.V-B, to evaluate the performance of the adaptive caching presented in Sec.V-C.

A. SIMULATION SETUP

The real-world YouTube data was collected using the YouTube API between the years 2013 to 2015 and consists of $J = 12,500$ YouTube videos. In the collected dataset, the viewcounts range from 10^2 to above 10^7. Therefore, to prevent the machine learning algorithms from biasing their prediction to only the videos with the highest viewcount, we scale the viewcount v_j to be on the log scale (i.e. if a video has 10^6 views then $v_j = 6$). All the content features are scaled to satisfy $x(m) \in [0, 1]$ for $m \in \{1, \ldots, M\}$. Note that we also collect the category $c \in \mathcal{C}$ of each YouTube video (e.g. "Entertainment", "Music", etc.), however, this information is not included into the feature set used to train the ELMs. In total there are 17 YouTube categories in the collected dataset. Each ELM is trained using an identical 10-fold cross validation method using the dataset $\mathcal{D}$, or in the case of the feature selection method $\mathcal{D}^*$. The trained ELMs are then used to compute both the video popularity $\hat{\mu}^i_j(t)$ (1), and categorical popularity $\hat{\mu}^{ic}(t)$ (10).

TABLE 2. Number of files in each YouTube category in the collected dataset.

Category	1	2	3	4	5	6
Number of files	230	2	1475	76	151	47
Category	7	8	9	10	11	12
Number of files	3774	300	406	1839	220	913
Category	13	14	15	16	17	
Number of files	69	126	10	2855	7	

For evaluating the performance of the adaptive caching scheme we require the network parameters, a method to generate user requests, and the cache initialization time of the MILP. Initially, the content is cached via the solution of the MILP (2) at simulation time slot $p = 1$ with the parameters $w_1 = 1$, $w_2 = 0.005$, $w_3 = 1$, and $z_0 = 0.1$. The topology of the network is provided in Fig. 5, and the parameters of the network are given by: $S = 9$ TB, $f_j = 500$ MB, $s_0 = 200$ GB, $r_i \in \{1, 2\}$, and Table 2 provides the size of all content in each video category. To generate user requests based on the real-world YouTube data, we use the following stochastic simulation.

$$\begin{aligned} \lambda^i &\sim \mathcal{U}[1, 10] \quad i \in \mathcal{V} \\ N^i_p &\sim \text{Poisson}(\lambda^i) \quad p \in \{1, \ldots, 50,000\} \\ F^i_q &\sim \text{Cat}(\boldsymbol{\mu}(t)) \quad q \in \{1, \ldots, N^i_p\} \end{aligned} \tag{19}$$

where N^i_p is the total number of requests at BS i at simulation time slot p, F^i_q is the video content that is requested at BS i at simulation time slot p by the q-th request. The categorical distribution $\text{Cat}(\boldsymbol{\mu}(t))$ is defined by the video popularity vector $\boldsymbol{\mu}(t) = [\mu_1(t), \mu_2(t), \ldots, \mu_J(t)]$ where $\mu^i_j(t)$ is defined in (1). The parameter $\boldsymbol{\mu}(t)$ is computed using the viewcount on day $t = 4$ ($\hat{v}^i_j(4)$). The content popularity is assumed to be equal at each BS. With the parameters in (19), each BS $i \in \mathcal{V}$ will receive on average between $50,000$ to $500,000$ content requests per day. To compute the latency parameters, d^{il} and d^{gi} of equation (2) we use the ndnSIM2.0 (an NS-3 based simulator) software [46]. ndnSIM's *Best Route* strategy is used to transfer content between BSs.

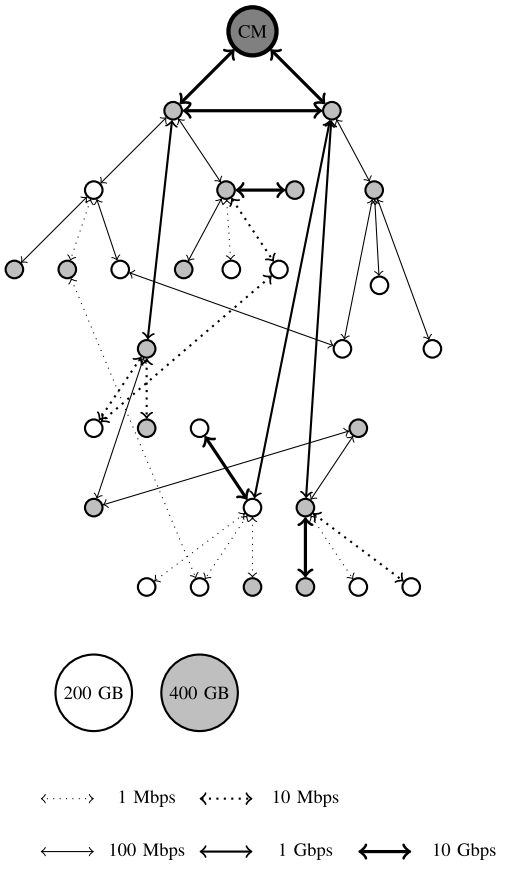

FIGURE 5. Schematic of the network. The circles with a solid black outline and light gray fill represent base stations having 400 GB storage size. Other base stations have 200 GB storage size. The associated communication links between the base stations are denoted by the connected arrows. CM is the content manager.

B. PERFORMANCE OF EXTREME LEARNING MACHINE FOR CACHING

In this section, using the real-world data from YouTube, we illustrate how the number of neurons of the ELM (11) and features can be selected using Algorithm 1 and Algorithm 2. Additionally we will illustrate how the ELM can be used to both predict the popularity of new videos, and estimate the popularity of published videos.

Fig. 6 illustrates the mean and variance of the specificity, false negative rate, false positive rate, sensitivity, and Gmean computed using 600 independently trained ELMs for each number of neurons. Using the SPSA (Algorithm 1) we found that an ELM with $L = 300$ provides sufficient accuracy for performing the content popularity estimation.

Having computed the optimal number of neurons, the next task is to select the video features which are most important for estimating the video viewcount. The performance of Algorithm 2 for selecting the YouTube features of the ELM is illustrated in Fig. 7(a) and 7(b). When using R^2, only 3 features are required to maintain a high level of performance for the ELM. These 3 features are the number of subscribers, contrast of the video thumbnail, and the overexposure of the video thumbnail. This illustrates that the title and keywords

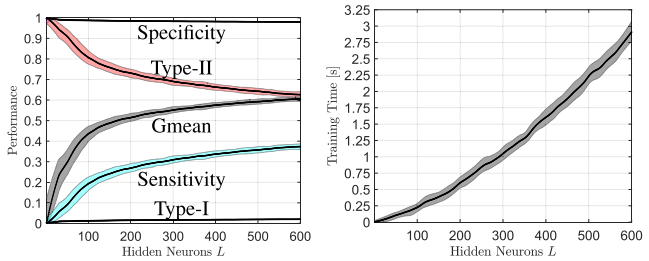

FIGURE 6. Performance of the ELM (11) for estimating YouTube video content popularity as a function of the number of neurons L in the ELM. The dataset used for this analysis is presented in Sec.IV-B. CM represents the cache manager which has the access to all the video files, and the other nodes represent the base stations used to serve user requests. (a) Hidden neurons L. (b) Hidden neurons L.

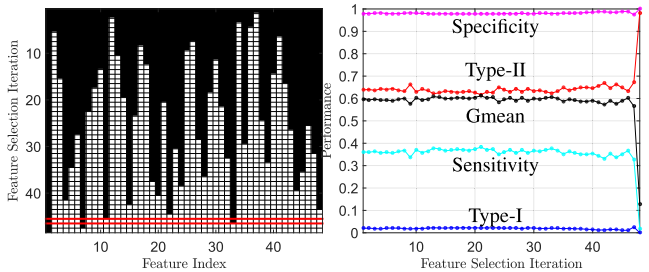

FIGURE 7. Performance of the feature selection Algorithm 2 when the R^2 coefficient of determination (Fig. 7(a) and 7(b)) are used as the similarity metric. (a) Feature index. (b) Feature selection iteration.

contribute negligibly to the popularity of YouTube videos in the dataset analysed when using the ELM.

Having optimized both the number of neurons of the ELM and the important features required for performing the popularity estimation, we now illustrate the performance of the ELM compared to several other machine learning methods. Fig. 8(a) provides a schematic of the ELM that can perform both prediction of new and published videos. The meta-data for a video is presented to the feature selection algorithm which constructs the video features $x_j \in \mathbb{R}^4$ which is composed of subscribers, contrast, overexposure, and previous day viewcount. Notice that $x_j(t)$ evolves per day t after the video is posted as new request statistics become available. The predicted viewcount on day t from the ELM is given by $\hat{v}_j(t)$.

As expected, with no request statistics available, the predicted viewcount from the ELM has a large variance as illustrated in Fig. 8(b) for $v_j(1)$. However in typical caching applications we are only interested in the top 10% of content in which case we can construct a binary popularity estimator using the output from the ELM by thresholding. For a video popularity threshold of $l_{\text{th}} = 10^{4.5}$ views there are 1379 popular videos and 11,121 unpopular videos. Table 3 provides the performance of the Binary ELM classifier and several other machine learning classifiers. Note that all classifiers were trained using the same dataset and on a standard desktop computer. As seen, the ELM has comparable performance to several popular classifiers and can be evaluated efficiently.

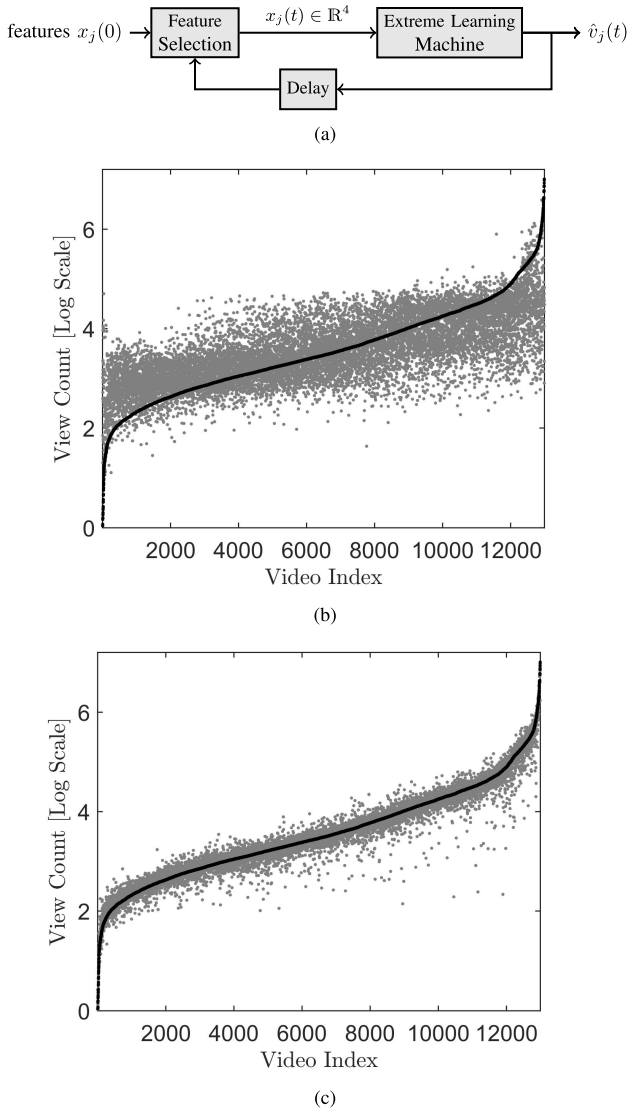

FIGURE 8. Real-World viewcount $v_j(t)$ (black dots) and numerically predicted viewcounts $\hat{v}_j(t)$ (grey dots) computed using the ELM (11). The ELM is trained using the YouTube dataset $\mathcal{D}$ as described in Sec.V. (a) Schematic of the Extreme Learning Machine for estimating the viewcount $v_j(t)$ of video $j \in \mathcal{J}$. $x_j(0) \in \mathbb{R}^M$ is the initial set of features of video $j \in \mathcal{J}$, $x_j(t) \in \mathbb{R}^4$ are the features used by the ELM to estimate the video viewcount $\hat{v}_j(t)$ on day t. (b) Viewcount on day 1. (c) Viewcount on day 4.

As the request statistics arrive the ELM can be used to make an accurate prediction of the viewcount dynamics as illustrated in Fig. 8(c) for the viewcount on day 4 (i.e. $v_j(4)$). Therefore a course estimate of the popularity of videos can be made using the ELM initially, then as request statistics arrive the ELM can be used to provide a high accuracy estimate of the next day popularity of videos.

C. PERFORMANCE OF THE CONTENT AND NETWORK AWARE CACHING SCHEME

This section illustrates the performance of the adaptive caching scheme presented in Sec.III. Specifically, the content downloading delay and cache hit ratio from the adaptive caching scheme are compared with the *most popular* and *random cache deployment* schemes.

TABLE 3. ELM performance comparison: TP (true positive), TN (true negative), and training times.

Method	TP	TN	Times (s)
Stochastic Gradient Boosting [47]	432	11509	5.41
Independent Component Regression [48]	769	11424	0.75
Generalized Linear Model [49]	769	11423	0.59
k-Nearest Neighbors [49]	1000	11524	24.55
Stacked AutoEncoder Deep Neural Network [50], [51]	959	11374	24.27
Boosted Tree [52]	1029	11419	16.32
Extreme Learning Machine	1067	11399	0.54

In the *most popular* caching scheme, each BS caches the most popular estimated categories (computed using request statistics) of the content until each base station's cache is full [6], [17]. The *random cache deployment* scheme accounts for content popularity (computed using content request statistics) and network parameters using an MILP which does not account for changes in the physical cache sizes at the BSs [21]. Additionally, the method in [21] incorporates a cache replacement scheme using the LRU (Least-Recently-Used) scheme. In this section, we compare the performance of the adaptive caching scheme with the schemes in [6], [17], and [21], with the predicted popularity from the ELM used in place of the request statistics, and the S3LRU used in place of the LRU cache replacement scheme.

To compute the optimal size of caches for the BSs, we solve the MILP problem (2). The results of the MILP are provided in Fig. 5 where the circles with a solid black outline and light gray fill represent BSs allocated with a 400 GB cache storage size. Other BSs are allocated with a 200 GB cache storage size. As seen in Fig. 5, the physical cache sizes in the network are heterogeneous. The MILP, based on the network topology, link capacity, routing strategy, and content popularity, optimally selects the physical cache sizes to use to reduce network energy consumption.

Fig. 9 shows the cumulative content downloading delay in the network vs. the simulation time. From Fig. 9, the adaptive caching scheme has the smallest cumulative content downloading delay compared with the *most popular* and *random cache deployment* schemes. This allows the adaptive caching scheme to increase the users' QoE in the network compared to these other caching schemes. The main reason the adaptive caching scheme outperforms the *most popular* and *random cache deployment* schemes is that the adaptive caching scheme considers adjacent BSs physical cache sizes and cached content to improve network performance. Comparing the performance of the *most popular* caching scheme and *random cache deployment* scheme, it is clear that methods which account for network parameters while making caching decisions will improve the users' QoE.

Fig. 10 plots the cumulative average cache hit ratio in the network vs. the simulation time. As seen in Fig. 10, the adaptive caching scheme performs better than the other two caching schemes. This performance improvement is due to

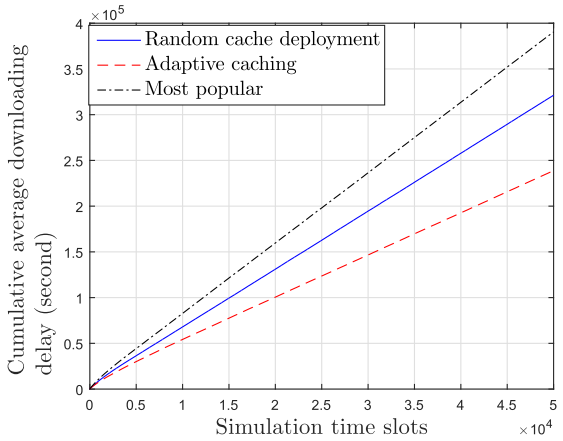

FIGURE 9. Cumulative average content downloading delay vs. simulation time. The figure illustrates that lower content downloading delay improves users' QoE.

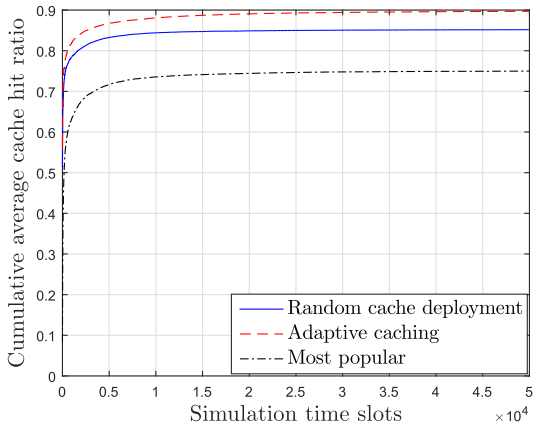

FIGURE 10. Cumulative average cache hit ratio in the network vs. simulation time. The figure illustrates that a higher cache hit ratio reduces the overall network traffic since fewer requests are served by transferring the content from the cache manager to the BS where the request originated.

the fact that the adaptive caching scheme takes into account network topology, content popularity and cache deployment in the formulation. Here, higher cache hit ratio in the network means, a higher number of requests are being served by the connected BSs or by the neighbour BSs. As can be seen from Fig. 10, cache hit ratio for adaptive caching scheme is 0.9. That means only 10% of the content requests are served by the content server while *random cache deployment* and *most popular* caching schemes account for 15% and 25%, respectively for the given simulation setup. Therefore, the adaptive caching scheme reduces network traffic since fewer requests are served from the content server.

VI. CONCLUSION

In this paper an adaptive caching scheme is presented that takes into account users' behavior and operating characteristics of the cellular network. The caching scheme uses an optimized extreme learning machine to estimate the popularity of content based on users' behaviour, features of the content, and request statistics from users as they become available. The features of the content are computed using a combination of human perception models and network parameters. The estimates are used in a mixed-integer linear program which takes into account the cellular network parameters (e.g., network topology, communication link, and routing strategy) to select where to cache content and also to provide storage recommendations to the network operator. The scheme is validated using real-world data from YouTube and the NS-3 simulator. In future work, we will consider how to optimize the update times of performing popularity estimation and cache updating based on both the content popularity and the technological network parameters.

ACKNOWLEDGMENT

The authors are grateful to Ngọc-Dũng Đào and Hang Zhang from Huawei Technologies Canada Co. Ltd. for useful insight and expertise that greatly assisted in the research of this paper.

REFERENCES

[1] Cisco Visual Networking Index: "Global Mobile Data Traffic Forecast Update 2015–2020," 2016.
[2] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.
[3] M. Gregori, J. Gómez-Vilardebó, J. Matamoros, and D. Gündüz, "Wireless content caching for small cell and D2D networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1222–1234, May 2016.
[4] N. Zhao, X. Liu, F. R. Yu, M. Li, and V. C. Leung, "Communications, caching, and computing oriented small cell networks with interference alignment," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 29–35, Sep. 2016.
[5] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, Oct. 2014.
[6] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.
[7] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, to be published.
[8] S. Andreev *et al.*, "Exploring synergy between communications, caching, and computing in 5G-grade deployments," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 60–69, Mar. 2016.
[9] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2995–3007, Jun. 2016.
[10] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3665–3677, Jul. 2014.
[11] B. Bharath, K. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1674–1686, Oct. 2016.
[12] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Mar. 2017.
[13] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jul. 2014, pp. 1897–1903.
[14] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. 11th Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2014, pp. 917–921.
[15] E. Baştuğ *et al.*, "Big data meets telcos: A proactive caching perspective," *J. Commun. Netw.*, vol. 17, no. 6, pp. 549–557, Dec. 2015.
[16] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.
[17] E. Bastug, M. Bennis, and M. Debbah, *Proactive Caching in 5G Small Cell Networks*. Hoboken, NJ, USA: Wiley, 2016, pp. 78–98.

[18] E. Bastug, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Proc. 13th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, 2015, pp. 161–166.
[19] E. Bastug, M. Bennis, and M. Debbah, "Anticipatory caching in small cell networks: A transfer learning approach," in *Proc. 1st KuVS Workshop Anticipatory Netw.*, 2014, pp. 1–9.
[20] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Netw.*, vol. 61, pp. 32–48, Jan. 2015.
[21] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2114–2127, Mar. 2016.
[22] J. Zhang, X. Zhang, and W. Wang, "Cache-enabled software defined heterogeneous networks for green and flexible 5G networks," *IEEE Access*, vol. 4, pp. 3591–3604, 2016.
[23] J. Liu, Q. Yang, and G. Simon, "Optimal and practical algorithms for implementing wireless CDN based on base stations," in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC Spring)*, Feb. 2016, pp. 1–5.
[24] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proc. 6th Int. Conf.*, 2010, pp. 4:1–4:12.
[25] J. Till, S. Engell, S. Panek, and O. Stursberg, "Empirical complexity analysis of a MILP-approach for optimization of hybrid systems," in *Proc. IFAC Conf. Anal. Design Hybrid Syst.*, 2003, pp. 129–134.
[26] K. Genova and V. Guliashki, "Linear integer programming methods and approaches–A survey," *J. Inf. Technol.*, vol. 11, no. 1, p. 1, Jan. 2011.
[27] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Oper. Res.*, vol. 46, no. 3, pp. 316–329, 1998.
[28] R. H. Bartels and G. H. Golub, "The simplex method of linear programming using lu decomposition," *Commun. ACM*, vol. 12, no. 5, pp. 266–268, 1969.
[29] F. A. Potra and S. J. Wright, "Interior-point methods," *J. Comput. Appl. Math.*, vol. 124, no. 1, pp. 281–302, 2000.
[30] Q. Huang, K. Birman, R. van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of Facebook photo caching," in *Proc. 24th ACM Symp. Oper. Syst. Principles*, 2013, pp. 167–181.
[31] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on mapreduce," *Neurocomputing*, vol. 102, pp. 52–58, Feb. 2013.
[32] A. Basu, S. Shuo, H. Zhou, M. H. Lim, and G.-B. Huang, "Silicon spiking neurons for hardware implementation of extreme learning machines," *Neurocomputing*, vol. 102, pp. 125–134, Feb. 2012.
[33] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, p. 386, 1958.
[34] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3460–3468, Oct. 2008.
[35] W. Huang *et al.*, "A semi-automatic approach to the segmentation of liver parenchyma from 3D CT images with extreme learning machine," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jun. 2012, pp. 3752–3755.
[36] J. Zhao, Z. Wang, and D. Park, "Online sequential extreme learning machine with forgetting mechanism," *Neurocomputing*, vol. 87, no. 2, pp. 79–89, 2012.
[37] Y. Ye, S. Squartini, and F. Piazza, "Online sequential extreme learning machine in nonstationary environments," *Neurocomputing*, vol. 116, no. 2, pp. 94–101, 2013.
[38] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.
[39] C. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in *Proc. 9th Int. Conf. Weblogs Social Media*, 2014, pp. 216–225.
[40] Q. Yu *et al.*, "Ensemble delta test-extreme learning machine (DT-ELM) for regression," *Neurocomputing*, vol. 129, pp. 153–158, Apr. 2014.
[41] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 21, no. 1, pp. 158–162, Jan. 2010.
[42] J. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation*, vol. 65. Hoboken, NJ, USA, Wiley, 2005.
[43] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, vol. 454. New York, NY, USA: Springer, 2012.
[44] U. Stańczyk and L. Jain, *Feature Selection for Data and Pattern Recognition*. Berlin, Germany: Springer, 2015.
[45] F. Benoît, M. van Heeswijk, Y. Miche, M. Verleysen, and A. Lendasse, "Feature selection for nonlinear models with extreme learning machines," *Neurocomputing*, vol. 102, pp. 111–124, Feb. 2013.
[46] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," NDN, Los Angeles, CA, USA, Tech. Rep. NDN-0028, Jan. 2015.
[47] J. H. Friedman, "Stochastic gradient boosting," *Comput. Statist. Data Anal.*, vol. 38, no. 4, pp. 367–378, 2002.
[48] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*, vol. 46. Hoboken, NJ, USA: Wiley, 2004.
[49] W. Venables and B. Ripley, *Modern Applied Statistics With S-PLUS*. New York, NY, USA: Springer, 2013.
[50] J. Hu, J. Zhang, C. Zhang, and J. Wang, "A new deep neural network based on a stack of single-hidden-layer feedforward neural networks with randomly fixed hidden neurons," *Neurocomputing*, vol. 171, pp. 63–72, Jan. 2015.
[51] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[52] R. Quinlan, *C4.5: Programs for Machine Learning*. Amsterdam, The Netherlands: Elsevier, 2014.

S. M. SHAHREAR TANZIL received the B.Sc. degree in electrical and electronics engineering from the Bangladesh University of Engineering and Technology, Bangladesh, in 2011, and the M.A.Sc. degree from The University of British Columbia, Vancouver, BC, Canada, in 2013, where he is currently pursuing the Ph.D. degree. He is a member of the Statistical Signal Processing Laboratory. His research interests include wireless networks and cloud computing.

WILLIAM HOILES received the M.A.Sc. degree from the Department of Engineering Science, Simon Fraser University, Burnaby, Canada, in 2012, and the Ph.D. degree from Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada, in 2015. He was a Post-Doctoral Researcher of Electrical Engineering with the University of California at Los Angeles, Los Angeles, CA, USA, in 2016. He is currently a Post-Doctoral Lecturer of Electrical and Computer Engineering with The University of British Columbia. His current research interests include social sensors and the bioelectronic interface.

VIKRAM KRISHNAMURTHY (S'90–M'91–SM'99–F'05) received the Ph.D. degree from Australian National University, Canberra, Australia, in 1992. From 2002 to 2016, he was a Professor and Canada Research Chair with The University of British Columbia, Canada. He is currently a Professor with the Department of Electrical and Computer Engineering, Cornell Tech and Cornell University. He is the author of the book *Partially Observed Markov Decision Processes* (Cambridge University Press, 2016). His research interests include statistical signal processing, computational game theory, and stochastic control in social networks. He served as a Distinguished Lecturer of the IEEE Signal Processing Society and an Editor-in-Chief of the IEEE JOURNAL ON SELECTED TOPICS IN SIGNAL PROCESSING. In 2013, he was awarded an Honorary Doctorate from the Royal Institute of Technology, Sweden.