Quality estimation for DASH clients by using Deep Recurrent Neural Networks

Bita Kheibari
Ege University
International Computer Institute
Izmir-Turkey
bita.kheibari@gmail.com

Müge Sayıt
Ege University
International Computer Institute
Izmir-Turkey
muge.sayit@ege.edu.tr

***Abstract*—Dynamic Adaptive Streaming over HTTP (DASH) is a technology designed to deliver video to the end-users in the most efficient way possible by providing the users to adapt their quality during streaming. In DASH architecture, the original content encoded into video streams in different qualities. As a protocol running over HTTP, the caches play an important role in DASH environment. Utilizing the cache capacity in these systems is an important problem where there are more than one encoded video files generated for each video content. In this paper, we propose a caching approach for DASH systems by predicting the future qualities of DASH clients. For the prediction, we use learning model, and the qualities that will be cached are determined by using this model. The learning model is designed using Recurrent Neural Networks (RNNs) and also Long Short Term Memory (LSTM) which is a special type of RNNs with default behavior of remembering information for long periods of time. We also utilize SDN technology to get some of the outputs for the learning algorithm. The simulation results show that predicting future qualities helps to reduce the underruns of the clients when cache storage is utilized.**

***Index Terms*—DASH Streaming, SDN, LSTM , caching**

I. Introduction

One of the most popular Internet applications, video streaming systems, adapt to quality regarding the various network conditions in order to provide the best Quality of Experience (QoE) under the constraint of available network resources. In these systems, the encoded video is sent over HTTP. Quality adaptation is provided by producing quality alternatives on the server side and by selecting different qualities over time on the client's side. The selection of the quality of the video is done by an algorithm which is called " rate adaptation algorithm ".

In HTTP adaptive video streaming systems, the same video file is encoded at various bitrates to provide quality alternatives, called representations. The small partitions of the representations, i.e. segments, enable the clients to send HTTP requests for downloading the selected segments. The information about segments and representations such as their timing, URL addresses, media characteristics like video resolution and bit rates are kept in a file called Media Presentation Description (MPD). MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard covers the main aspects of such systems related to storage, transmission and parsing of the MPD file. The rate adaptation algorithm of the clients are not within the scope of DASH standard.

Software Defined Networking (SDN) is a new generation of network technology which allows separating the control and data plane. SDN reduces hardware dependency and increases software and network intelligence capabilities. In SDN domains, the network is managed by the controller via the communication protocol between the control plane and the data plane such as OpenFlow [1]. SDN technology can be used to implement application specific approaches at the network layer as well as providing abstracted network conditions information to the applications.

In order to increase the performance of DASH systems, network elements such as SDN controller or DASH aware middleboxes can provide help to the clients. MPEG group's recent standard Server and Network Assisted DASH (SAND) developed by considering these and introduces DASH aware network elements can be given as example for this purpose [2]. Caches are one of the main elements in video streaming systems. Therefore, the caching approaches using DASH specific knowledge can provide improvement of QoE.

In this paper, we propose an approach for deciding the representations that are cached for improving the performance of the DASH clients running over an SDN domain. The cache implementation runs as a northbound application at the SDN controller. For deciding which qualities will be cached, Supervised Learning with RNNs is used. The clients communicates with the controller periodically to provide internal parameters, which, in turn are used as inputs to the learning model.

There are a lot of caching strategies, which address to determine the video files to be cached, which has been proposed in the literature and with these approaches cache hit ratio is remarkably increased [3]. In DASH systems, since there are more than one representation for each video content, the selection of the representations is for caching another problem. Different from the existing work, in this study we focus on selecting the quality alternatives, i.e. representations, to be cached, rather than the video files. Our approach can be easily combined with the approaches proposed for determining video files to be cached. The selection of the video files and selection of the representation of a video file have different characteristics. While user preferences play a role for selecting the video files, the selection of the video quality is done by the client's software. Different from the literature, the proposed learning model for predicting the future representations of DASH clients considers the type of the rate adaptation algorithm and utilizes SDN technology. However, the details of the rate adaptation algorithm is not known, the learning algorithm infers the actions of the client and predicted representations are cached. We show that the caching strategy by utilizing the proposed learning model can reduce the underruns on the client's side. We also implement basic schemes for deciding the future representations and provide comparative results about caching strategies utilizing future representation predictions.

The rest of the paper is organized as follows: Section 2 will review related work of the study. In sections 3 and 4 an overview of the proposed architecture and the relevant analysis of our RNN model will be explained in detail respectively. The conclusion of the study has been provided in section 5 with the following references.

II. Related Works

In this section, we provide an overview of the most relevant works, improving the QoE by reducing the re-buffering time for the video being streamed.

A. Recurrent Neural Networks

The developments of softwarized and autonomous network technologies in recent years lead researchers to develop machine learning based network solutions to increase the performance of the network applications. There are three types of learning: Supervised, Unsupervised and Reinforcement learning. In Supervised learning, agents learn from the feed of labeled data. The learning works by explicitly given the inputs and the outputs being based on the inputs. Unsupervised learning models run without having pre-knowledge or a guidance, and the data is not labeled accordingly. The model finds undetermined patterns in order to make predictions about the output. On the other hand, in Reinforcement learning, an agent interacts with its environment and discovers its characteristics based on the received punishments or rewards.

Neural Networks are set of algorithms which are akin to the human brain and are devised to detect patterns. A cluster of neurons collaborating so closely with each other in order to resolve an issue, build a concept which is referred to an artificial neural network.

Recurrent Neural Networks (RNN) is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past computations. After generating the output, it is copied and sent back into the recurrent network. To make a decision, it considers the current input and the output that it has learned from the previous input. In other neural networks, all the inputs are independent of each other but in RNN, all the inputs are related to each other. In this study, we use supervised approach by utilizing a model with RNN.

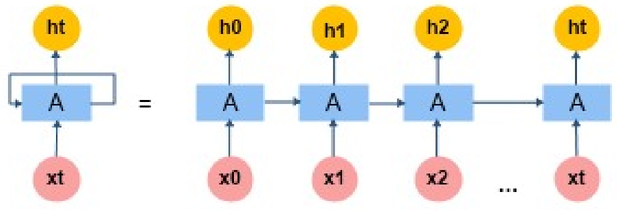


Fig. 1: An Unrolled Recurrent Neural Network.

As shown in Fig. 1, in the first step, the model takes the X(0) from the sequence of inputs and then computes and generates the h(0) as output which together with X(1) is the input for the next step. In fact, the h(0) and X(1) are the input for next(second) step. In a similar way, h(1) from the next (second) step with X(2) are the inputs for next (third) step and so on. In this way, it keeps remembering the context while training. ”A” is the arithmetic part of each neurons, which calculates the output based on our input parameters.

The formula for the current state is :

$$h_t = f(h_{t-1}, X_t)$$

To remember past data memory easier, Long Short-Term Memory (LSTM) networks are applied. Long Short-Term Memory (LSTM) is a specific RNN architecture that was designed to model temporal sequences and their long-range dependencies are more accurate than conventional RNNs. The vanishing gradient problem of RNN is rectified by utilizing LSTM which is proper to organize, process and anticipate time sequences, time lags of indistinct duration. It keeps fixing the model by using back-propagation. In an LSTM network, three gates are present:

- **Input gate**: Explores what value from input has to be utilized to modify the memory.
- **Forget gate**: Probes what details to be discarded from the block.
- **Output gate**: Output is determined based on the input and the memory of the block.

In neural networks the responsibility of a neuron is to provide an output by applying a function on the inputs provided. The function used in the neuron is generally termed as an Activation function. The most commonly used activation function in deep learning models is Rectified Linear Unit (Relu) [4] . The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back [5]. So, it can be written as ***f(x)=max(0, x)***.

$$Relu(x) = \begin{cases} 0 & x < 0 \\ x & x >= 0 \end{cases}$$

As it can be seen from the formula, this activation function doesn't have high complexity and it's easy to be implemented and used. The learning process is fast and easy, which is an important aspect of a learning algorithm to be used in a highly dynamic environment such as DASH.

B. *Utilizing Learning Models for DASH Systems*

One class of the machine learning based applications designed for DASH systems is based on the idea of developing rate adaptation algorithms with ML. In [6], the authors presented a framework (D-DASH) that combines deep learning and reinforcement learning techniques to improve the quality of experience of DASH. Different learning architectures are proposed by combining feed-forward and recurrent deep neural networks. LSTM is a good alternative to predict the performance of the DASH systems due to its characteristics depending on time series. Another rate adaptation algorithm, maximizing a chunk wise subjective QoE model by utilizing it as the reward function in reinforcement learning (RL) is proposed in [7]. In [8], the authors proposed an LSTM based model in order to estimate optimal quality for the further segments. This model is used by the DASH clients within the rate adaptation algorithm and the purpose of the model is to define the quality under the constraint of internal parameters. Hence, the model, the input, outputs and the aim of the study is different from this current work. In general, learning models are developed for the rate adaptation algorithms used in the client's software, and differ from our study since our approach is run by the cache.

The other class of the machine learning based approach developed for DASH systems focus on the improvements on the network side or on the network elements. In that sense, the performance of DASH systems can be increased by utilizing SDN and machine learning techniques. In [9], by using Multi-Criteria Decision Making (MCDM) method, an approach that aims to improve the quality of the buffered video on the client's side is proposed. The authors utilize SDN for deciding the weights of the MCDM method according to this SDN controller runs a machine learning algorithm by using its knowledge about current network conditions as an input of the learning algorithm. An SDN-based architecture framework that targets to optimize the QoE for video streaming in SDN networks for DASH is designed in [10]. A learning model is run by using the information at the controller, and the output is the quality that provides the best QoE. In [11], authors presented a machine learning model that by using the client-side features included the current bitrate, the current bandwidth, and the current buffer size could distinguish the target quality level. To reach this aim they also used Supervised classification. Their model could be successfully utilized within a DASH client for selecting the optimal quality for each client based on the client's current capabilities. Caching strategies are not addressed in any of these studies.

Using the prediction mechanisms for proactive caching is a problem which is widely studied. In general, the studies proposed a prediction model for determining the popularity of the videos so that the popular contents can be cached in advance by analyzing data such as historical user demands [12] or content quality [3]. The prediction of the future segments of DASH clients has been getting the researchers' attention because it might help to increase cache hits by caching future segments, which in turn provide to use cache storage more effectively. As one of the premier studies based on this idea, in [13], the authors propose an approach for predicting future segments, which determines the future quality selections by using bandwidth estimations and the knowledge of the rate adaptation algorithm. In [14], the authors provide a proactive caching approach having Markov property. The model considers the network conditions to predict the segment which will be requested in the future. However, it is assumed that the cache knows the client's rate adaptation as well as the client's actions and the details rate adaptation algorithm might not be known even by the users. Different from these studies in the literature, in this current work, we develop an approach based on LSTM, which aims to predict the future segments of the clients without the necessity of having knowledge about the rate adaptation algorithm run by the clients.

There are also a few studies proposing to select representations for cache prefetching without the need for having knowledge about rate adaptation algorithm of the clients. For the selection of the representations to be prefetched, the authors propose to cache representations according to the outputs of an optimization model in [15]. This MILP based model provides the representations maximizing the QoE, however, the rate adaptation algorithms of the clients may not select the representations maximizing QoE due to the reasons such as internal parameters and the lack of the knowledge about real network capacity. In [16], the authors proposed a learning based edge-cache platform. Their system collects clients' information such as throughput, obtained QoE values, requested video type and then prefetching is done by considering QoE gain. In the current study, we simply focus on predicting the client's representation selection rather than focusing on QoE because each rate adaptation algorithm has different policy to maximize QoE. Although QoE is not directly addressed in the model, the proposed approach helps to increase QoE on client's side due to the minimization of the latency between the resource and the client.

III. The Proposed System Architecture

A. *The Prediction of Future Segments with RNN*

In this study, we consider the clients connect to the SDN domain which is managed by a controller. The controller, as a point having the information about network such as the the end-to-end paths within its domain and real-time traffic volume over these paths, has a cache implementation running as an application.

After the clients connected to the network and started downloading the segments, they periodically send the values of buffer fullness, calculated throughput, segment size, segment ID and selected qualities to the controller after downloading each segment. When the learning algorithm is at the dataset collection stage, the controller gathers this information sent by the clients. These parameters are used as the inputs to the learning model. The reason for why we use segment ID as an input is that, using this value helps the learning model to infer initial waiting time. In general, the first few segments are buffered during the initial waiting time and the quality of these segments are minimum regardless of the estimated throughput in order to minimize the initial waiting period. The learning model utilizes the segment ID value information in order to predict segment quality at the beginning of the stream.

As explained in the previous sections, the learning model is based on Recurrent Neural Network. Because of the special properties of LSTM, which is designed to store and access information better than the general-purposed neural networks, the model is built by using LSTM. Unlike the traditional recurrent neural networks in which the content is rewritten at each time step, in a LSTM recurrent neural network, the network is able to decide on saving the current memory through the introduced gateways. Such neural networks are particularly useful for processing the sequential data in which each neuron or processing unit is able to maintain the internal state or memory to retain information related to the previous input. This feature is especially important in various functions of the sequential data, which fits the selected qualities in DASH system because the consecutive segments correlated to each other due to the similar throughput or buffer level values within consecutive request times.

The Recurrent Neural Network Model used in this study is made up of 3 hidden layers. The first hidden layer has 30 neurons while the second and third layers have 40 neurons, respectively. Consequently the next output representation is predicted by using this model. We used ReLu activation function, which is the most used activation function in the studies recently, to measure the output for each layer. The illustration of the designed RNN is shown in Fig. 2.

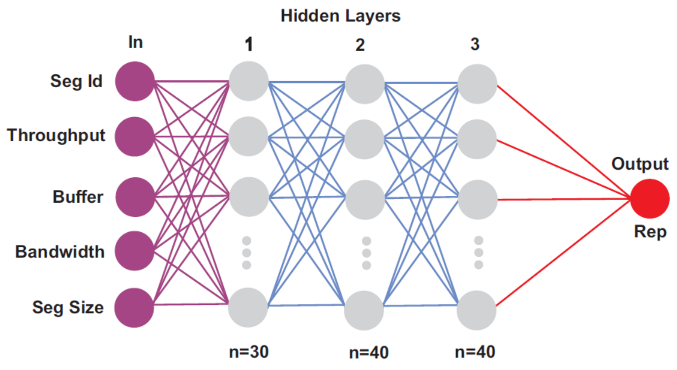


Fig. 2: The Neural Network Model.

In the training phase, the parameters of the learning model is finalized. After that, the parameters are started to be used by the cache application. However, at this stage, some inputs of the learning model such as throughput and current buffer level are not known by the controller since the model predicts future qualities. At this stage, the clients continue to send internal parameters to the controller. The inputs of the learning model which are going to be used after it's trained are client's expected throughput, segment ID and client's buffer fullness value. The output is the quality of the future segments. Expected throughput is calculated by dividing the related segment size to the available bandwidth. Available bandwidth is measured by the basic network functions within the SDN controller. Future segment sizes are obtained from the MPD file. Note that, this cache application can be an application from a video streaming company. Hence, MPD file can be provided by the company to the cache application.

B. *Cache Implementation and SDN Architecture*

In network architecture designed in this study, SDN controller has cache application as the northbound applications as well as basic network functions providing network related information for these applications. The illustration of the SDN based architecture is given in Fig. 3. The clients connected to SDN domain and the SDN controller architecture are shown in the figure. Basic network service functions of the controller are responsible for fundamental management functions of the network, such as measuring the traffic volume, tracking the host connections or sending the necessary commands to the switches. Our proposed cache application is run as one of the business applications, and at the controller there might be other business applications from the 3rd parties as well. The controller communicates with these business application by using its North-bound Interface (NBI). As also seen from the figure, the messages carrying the client's information are directed to the learning module if the system is in the training phase and are directed to cache application to be used as inputs for the learning model within the cache.

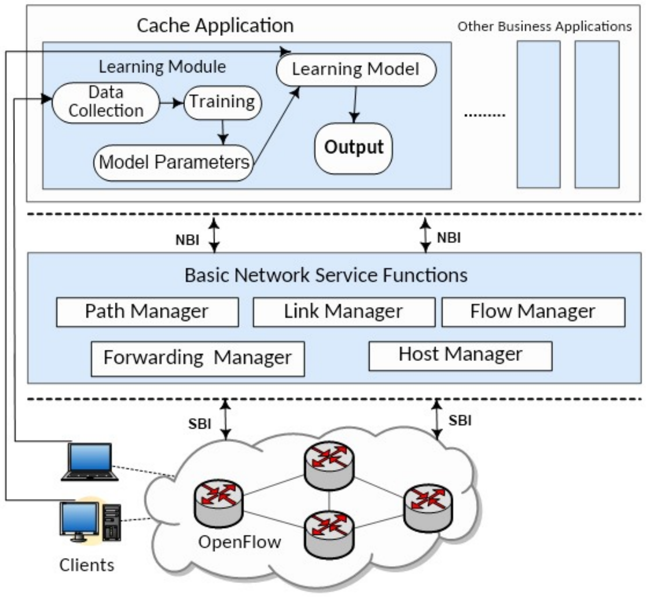


Fig. 3: General Illustration of the Proposed Architecture.

If the rate adaptation algorithm is known for the newly connected client, the cache implementation starts utilizing the learning model. Note that, the knowledge about the rate adaptation algorithm is not related to how the algorithm works, it's the type of the client's software such as DASH-JS [17] or Apple's HLS. The clients periodically send information about their buffer levels and measured throughput values as well as the current segment ID. These information is used as input to the learning model and the output gives the prediction for the next segment quality. The segments with the predicted quality is downloaded from the original server. Hence, this reduces the cache miss ratio. However, because the time required for caching the predicted segment equals to the sum of the running time of learning algorithm and the time for downloading, the learning model can also be used for the prediction of the further segments.

IV. Performance Evaluation

A. Dataset Preparation and Simulation Parameters

As the first step, we generated a dataset for training the learning model. The rate adaptation algorithms are throughput based, buffer based or hybrid based by using both throughput and buffer values. If a rate adaptation algorithm selects the next representation by considering only throughput or buffer level, its selections are based on one-to-one mapping between some predetermined range of the related values and the representation, hence it is easier to predict the selections after detecting its nature. In order to measure the performance of the proposed learning model by using a more complicated rate adaptation algorithm, we select a hybrid based algorithm. For this purpose, SARA (Segment Aware Rate Adaptation Algorithm), a well-known rate adaptation algorithm proposed in the literature is selected [18]. SARA uses estimated throughput and buffer values in order to select the representation. In this algorithm, the buffer is divided into smaller parts. There are different selection strategies for each part that follow a rule set when the buffer reaches the desired range. To predict the representations that will be selected by this algorithm, the learning model should infer that it uses both throughput and buffer values and acts regarding the buffer ranges.

Dataset is generated by running tests with SARA algorithm and collecting its outputs, i.e. representation selections. It should be noted that, the higher the number of tests, the higher the learning accuracy and subsequently getting more accurate results, so it's still room for some improvements of the learning algorithm by producing a better data set. The tests are conducted on Mininet platform. Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and the topology consists of OpenFlow enabled switches which communicates with the SDN controller [20].

In this study, the topology used in the experiments consists of a cache, an original server, four switches and two clients as shown in Fig. 4. Since there are more than one available paths in the topology, the controller selects the streaming path for the clients by considering the available bandwidth of the paths. However, the clients always use the same streaming paths. The reason for that is to provide the learning algorithm to learn the results when there are competing TCP flows which affect the selected representations [19]. After joining the network, the clients connect to the cache and requests a video. Our test video is "Big-Buck-Bunny" which contains 299 segments and each segment's length is 2 seconds. There are 6 representations available for the same video, whose bitrates are 2500, 3000, 4000, 5000, 6000 and 8000 (Kbps).

In order to generate the dataset, 100 series of tests were performed with different network conditions, in which available bandwidths of the links changes dynamically. Hence, based on the different bandwidth settings, the clients have different buffer values and request different representations during streaming, which helps the learning model to train properly. Link bandwidths are set as in the range of 1 to 8 Mbps and are dynamically changed every 30 seconds in random manner. For the training phase, all representations are cached for simplicity. After training with the prepared dataset, the Recurrent Neural Network is used for the prediction of the future segments for newly connected clients. In the next section, we give the performance results of the learning algorithm when it is used for prediction.

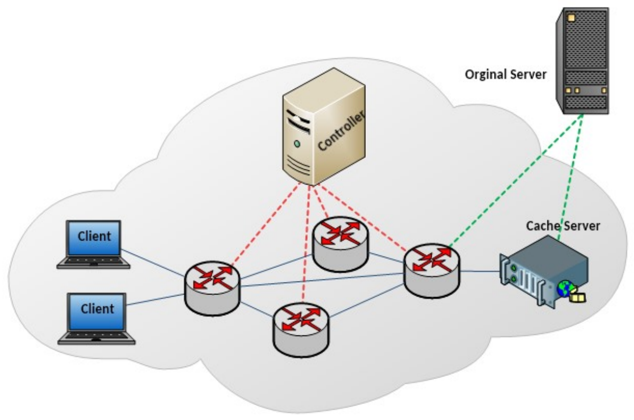


Fig. 4: Network Topology.

B. Performance Evaluation and Experimental Results

In order to measure the prediction performance of the learning algorithm, we run the algorithm for predicting segments selection of the newly connected clients. In this first set of tests, we measure the accuracy of the predictions of the algorithm. In addition to these measurements, to present the advantages of the representation selection for caching, we present the performance results in terms of client's experience when the proposed learning model used for caching.

In the first set of tests, after clients connect to the network, they periodically send their buffer values, measured throughput, segment ID and segment bitrate. When the controller receives these parameters from a client, it runs the learning algorithm to estimate the future segment for that client. All tests whose results are given in this section are conducted five times and the results are averaged.

According to the first stage of the tests, we compared the predictions of the learning algorithm and the client's selections. The learning algorithm is used for the predicting of the representation selection for the next $(x+1)^{st}$, $(x+2)^{nd}$, $(x+3)^{rd}$, $(x+4)^{th}$ and $(x+5)^{th}$ segments after a client sends the information related to its x^{th} segment selection. The predictions were compared with the client's representation selections for the segments from $(x+1)^{st}$ to $(x+5)^{th}$. The results are given in Fig. 5 and Fig. 6. We provide two different graphs for the first and the second client because even if the clients share the same streaming paths, their individual observation may differ due to the effects of TCP congestion algorithm. Hence, we present the prediction accuracy in order to show the prediction performance for comparing it to the selection of the clients having different partial knowledge of the same network conditions. In the figure, $(x+n)^{th}$ prediction refers to the prediction of $(x+n)^{th}$ segment when the parameters related to x^{th} segment is provided. The differences refer to the difference between the representation predicted by the learning algorithm and the representation requested by the client. When we examine the accuracy of the predictions, we observe that the prediction for the next segment and for the next 5^{th} segment is 70% and 65%, respectively.

In addition to these graphs, we also give the comparison of the selecting representations as a function time for a specific test. In Fig. 7, the compared values of predicted and real values for representation selection are shown for the next five segments. As it can be seen from the figure, in most cases, very accurate estimates have been made, but sometimes the predictions are quite different from the original requested representations. The first reason of this is that the future network conditions are not available in the dataset and the model has to estimate the future conditions which could be sometimes far from the expected conditions. The second reason is the sudden changes in bandwidth and sudden buffer level changes on the client's side. Because buffer level value is an input of the learning model and current buffer level values are used for the prediction of future representations, it leads to wrong predictions.

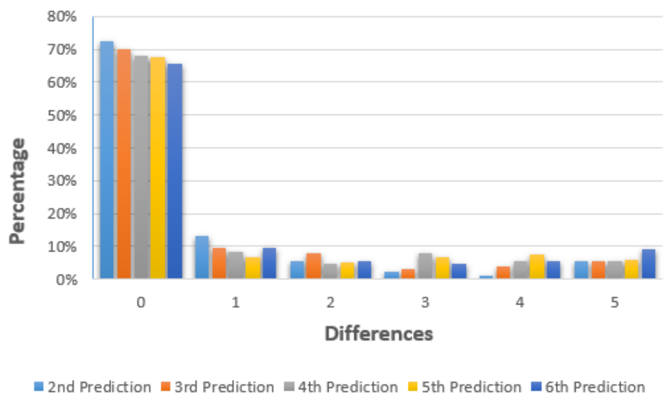


Fig. 5: The Prediction Accuracy Distribution of the Learning Model for Client 1.

After observing the prediction accuracy of the learning algorithm, in order to see its effects on the QoE, we run

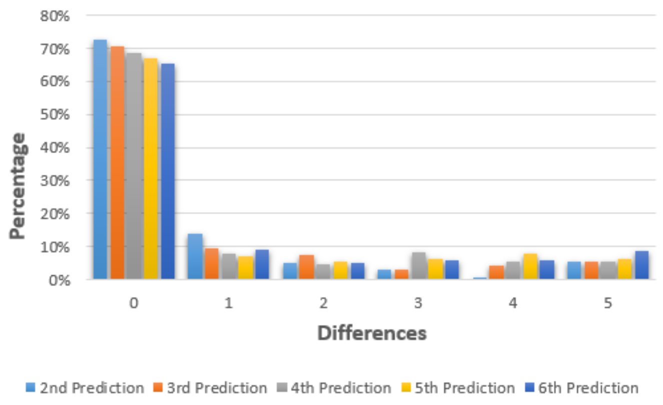
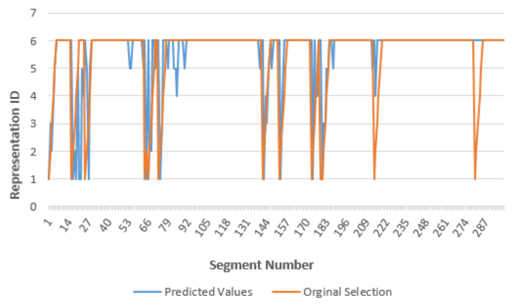


Fig. 6: The Prediction Accuracy Distribution of the Learning Model for Client 2.

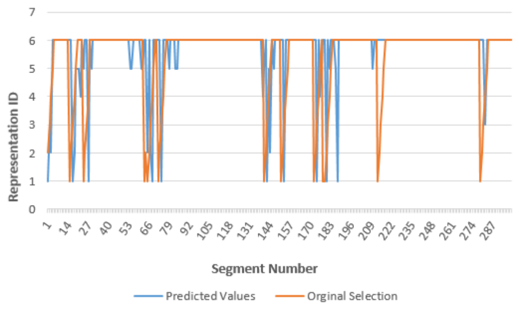
additional tests on Mininet. In these tests, the cache prefetches the representations based on the output of the algorithm. According to the observation on prediction accuracy, we consider the optimal approach as caching the predicted representations as well as next higher and next lower quality representations. Therefore, the cache only prefetches 3 representations out of 6 representations. This approach provides increase in the hit ratio compared to the case which only predicted representation is cached. We consider to cache for the next five segments; therefore, there will be sufficient amount of time for SDN controller to collect the clients' metrics, to run the algorithm and to prefetch the segments.

For the performance evaluation of the proposed learning based caching approach, we also implemented three caching strategies and compared the results. Since we focus on which representation to cache in this study, we also implemented Most Recently Used (MRU) and Most Frequently Used (MFU) strategies. For the implementation with these strategies, the clients' selections are collected periodically. While MRU caches the recent representations for the next 5 segments, MFU caches the mostly selected representation for the next 5 segments. In addition to that, we provide results for random caching.

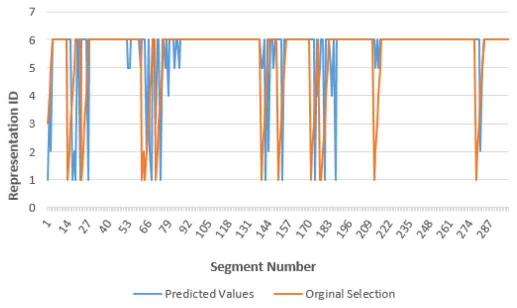
In Table I, the QoE parameters related to the underrun metrics are given for the four strategies. In the table, total underrun represents the averaged underrun values observed in the clients in one session. While max underrun represents the max underrun value observed in a client, min underrun value is the min value of the underrun values observed at the clients. The test results are obtained under the same network conditions for four strategies. As it can be seen from the table, random has the highest total underrun value while the learning algorithm based caching has the lowest total underrun values. The obtained results show that, representation prediction provides the decrease in undderrun. Furthermore, the learning based approach can provide up to 51% decrease in the underrun values when compared to random strategy.



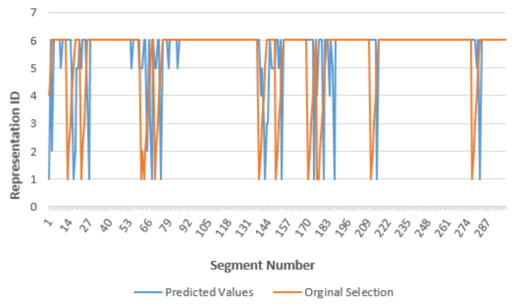
(a) First Prediction



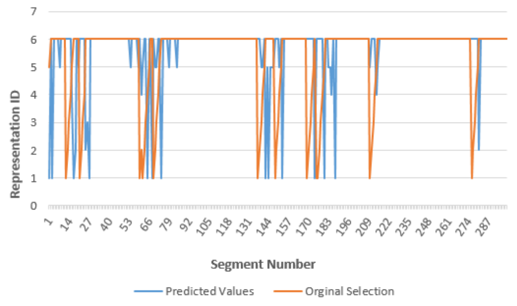
(b) Second Prediction



(c) Third Prediction



(d) Fourth Prediction



(e) Fifth Prediction

Fig. 7: Comparison of the predicted and real representation selections for the next five segments.

TABLE I: Underrun Metrics

Underrun Metrics	*Clients*	*Random*	*MFU*	*MRU*	*Learning*
Total Underrun	Client 1	180_S	125_S	104_S	93_S
	Client 2	162_S	116_S	98_S	92_S
Max Underrun	Client 1	12_S	11_S	15_S	14_S
	Client 2	15_S	13_S	12_S	15_S
Min Underrun	Client 1	5_S	6_S	6_S	6_S
	Client 2	6_S	5_S	7_S	6_S
Underrun Count	Client 1	21	15	11	11
	Client 2	19	14	10	10

V. CONCLUSION

The main purpose of this article is to use the facilities and benefits of learning neural networks in order to predict the most appropriate video quality representation and consequently increase the QoE. To achieve our goal, we used deep recurrent neural networks, which is a kind of supervised learning. In supervised learning, a set of prior information is required, and we used the SARA algorithm to generate a data set. These algorithms offer an appropriate representation by measuring series of criteria such as buffer occupancy, current bandwidth, and previous segment size. We designed a model using deep recurrent neural networks and since our data is sequential data we used LSTM networks. The parameters of the learning model were tuned by using the tests under various network circumstances and analyzing the results. The simulation results show that representation prediction can help to reduce underruns on the client side.

In the future work, we plan to measure the performance of the proposed caching approach by also using different rate adaptation algorithms. We plan to enhance the proposed learning model by adding future buffer level estimations to increase the prediction accuracy.

ACKNOWLEDGMENT

This work is funded by the Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 115E449.

REFERENCES

[1] Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. White paper , Palo Alto, CA, USA, Open Networking Foundation; 2012.
[2] E. Thomas, M.O. van Deventer, T. Stockhammer, A.C. Begen, M. Champel, O. Oyman, ”Applications and deployments of server and network assisted DASH (SAND)”, IET Conference Proceedings, 2016.
[3] H. S. Goian, O. Y. Al-Jarrah, S. Muhaidat, Y. Al-Hammadi, P. Yoo and M. Dianati, ”Popularity-Based Video Caching Techniques for Cache-Enabled Networks: A Survey,” in IEEE Access, vol. 7, pp. 27699-27719, 2019.
[4] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.
[5] Jeff Heaton , Deep Learning and Neural Networks , December 31, 2015/
[6] Gadaleta, M., Chiariotti, F., Rossi, M., Zanella, A. (2017). D-DASH: A deep Q-learning framework for DASH video streaming. IEEE Transactions on Cognitive Communications and Networking, 3(4), 703-718.
[7] Liu, J., Tao, X., Lu, J. (2018). QoE-oriented rate adaptation for DASH with enhanced deep Q-learning. IEEE Access, 7, 8454-8469.
[8] A. Lekharu, S. Kumar, A. Sur and A. Sarkar, ”A QoE aware LSTM based bit-rate prediction model for DASH video,” 2018 10th International Conference on Communication Systems Networks (COMSNETS), Bengaluru, 2018, pp. 392-395.

[9] Ozcan, S. G., Sayit, M. (2019, February). Improving the QoE of DASH over SDN: A MCDM Method with an Intelligent Approach. In 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN) (pp. 100-105).

[10] Abar, T., Letaifa, A. B., Elasmi, S. (2018, May). Enhancing QoE based on machine learning and DASH in SDN networks. In 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA) (pp. 258-263).

[11] Alzahrani, I. R., Ramzan, N., Katsigiannis, S., Amira, A. (2018). Use of Machine Learning for Rate Adaptation in MPEG-DASH for Quality of Experience Improvement. In 5th International Symposium on Data Mining Applications (pp. 3-11). Springer, Cham.

[12] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang and X. Shen, ”Content Popularity Prediction Towards Location-Aware Mobile Edge Caching,” in IEEE Transactions on Multimedia, vol. 21, no. 4, pp. 915-929, April 2019.

[13] P. Juluri and D. Medhi, ”Cache’n DASH: Efficient Caching for DASH”, ACM Conference on Special Interest Group on Data Communication (SIGCOMM), NY, USA, 2015, pp. 599–600.

[14] R. Coutinho, F. Chiariotti, D. Zucchetto and A. Zanella, ”Just-in-time proactive caching for DASH video streaming,” 2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Capri, 2018, pp. 1-6.

[15] A. Araldo, F. Martignon and D. Rossi, ”Representation selection problem: Optimizing video delivery through caching,” 2016 IFIP Networking Conference (IFIP Networking) and Workshops, Vienna, 2016, pp. 323-331.

[16] W. Shi, Q. Li, C. Wang, G. Shen, W. Li, Y. Wu, and Y. Jiang, ”LEAP: learning-based smart edge with caching and prefetching for adaptive video streaming”. In Proceedings of the International Symposium on Quality of Service (IWQoS), NY, USA, 2019, pp. 1–10.

[17] B. Rainer, S. Lederer, C. Müller and C. Timmerer, ”A seamless Web integration of adaptive HTTP streaming”, EUSIPCO, 2012.

[18] Juluri, P., Tamarapalli, V., Medhi, D. (2015, June). SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP. In 2015 IEEE International Conference on Communication Workshop (ICCW) (pp. 1765-1770).

[19] C. Cetinkaya, K. Herguner, C. Hellge, M. Sayit, ”Segment-aware dynamic routing for DASH flows over software-defined networks”, Int J Network Mgmt., 30:e2102, 2020.

[20] http://mininet.org