LOW-COMPLEXITY REINFORCEMENT LEARNING FOR DELAY-SENSITIVE COMPRESSION IN NETWORKED VIDEO STREAM MINING

Xiaoqing Zhu

Advanced Architecture and Research
Cisco Systems Inc.
San Jose, CA, USA

Cuiling Lan[†] *and Mihaela van der Schaar*[‡]

[†]Xidian Univesity, Xi'an, China
[‡]University of California at Los Angeles,
Los Angeles, CA, USA

ABSTRACT

In networked video stream mining systems, real-time video contents are captured remotely and, subsequently, encoded and transmitted over bandwidth-constrained networks for classification at the receiver. One key task at the encoder is to adapt its compression on the fly based on time-varying network bandwidth and video characteristics — while attaining low delay and high classification accuracy. In this paper, we formalize the decision at the encoder side as an infinite horizon Markov Decision Process (MDP). We employ low-complexity, model-free reinforcement learning schemes to solve this problem efficiently under dynamic and unknown environment. Our proposed scheme adopts the technique of virtual experience (VE) update to drastically speed up convergence over conventional Q-learning, allowing the encoder to react to abrupt network changes on the order of minutes, instead of hours. In comparison to myopic optimization, it consistently achieves higher overall reward and lower sending delay under various network conditions.

Index Terms— networked video stream mining, Markov decision process (MDP), reinforcement learning Q-learning with virtual experience (VE) update, human action recognition.

1. INTRODUCTION

Video stream mining stays at the core of many computer vision tasks, such as human action recognition [1] [2], road traffic analysis [3]. and event detection [4] [5]. As cameras become cheaper, better, more prevalent and more connected, they give rise to a wide range of novel applications for *networked* video stream mining. Such a system can capture, transmit, aggregate, and process large volumes of video content on the fly. Typically, video streams are captured at distributed sites. They are subsequently transmitted over best- effort networks to a central location, such as data center in the cloud, to furnish further mining tasks.

One key challenge in networked video stream mining is encoder adaptation. To save on transmission costs and to alleviate uplink network congestion, it is desirable to heavily compress the captured video content. On the other hand, the compressed video stream should retain sufficient information for effective mining at the receiver. In many classification-based mining systems, this translates into high classification accuracy of the compressed video streams. The trade-off between compression efficiency and mining effectiveness is further complicated by the requirements of low delay in transmission and processing, so as to support the detection of live events. In addition, the sender needs to rapidly *adapt* its encoding strategy to unknown and dynamic network and video conditions.

This paper explores the performance trade-off between available network resource, classification accuracy, and transport delay via encoder adaptation. We formalize the encoder adaptation decision as an infinite horizon Markov decision process (MDP). Our formulation takes sending buffer size as system state, considers various encoding quantization parameters (QPs) as candidate actions, and strives to maximize the long-term combining reward reflecting both classification accuracy and sending buffer size. We show that the proposed MDP problem can be solved efficiently by low-complexity, model-free reinforcement learning solutions, *without* explicit knowledge of video content characteristics and network bandwidth variations.

In addition to classic reinforcement learning techniques such as myopic optimization and conventional Q-learning, we propose an improved Q-learning scheme with virtual experience (VE) update. The proposed schemes are evaluated using the application example of human action recognition with the KTH dataset [6]. Experimental results show that both schemes based on Q-learning (with or without VE) consistently achieves higher overall rewards at lower sending delays than myopic optimization. The performance gain is more pronounced when network bandwidth is scarce, with as much as 90% of delay reduction from myopic optimization. Moreover, Q-learning with VE is shown to significantly speed up the convergence process, allowing the sender to react swiftly to abrupt changes in network bandwidth on the order of minutes, instead of hours.

The rest of the paper is organized as follows. After a brief survey of related work in the next section, Section 3 provides an architecture overview of the networked stream mining system. Section 4 describes the feature extraction procedure and the multi-class support vector machine (SVM) classifier used in human action recognition. In Sections 5 and 6, we present our MDP-based problem formulation and three reinforcement learning schemes. Section 7 evaluates the performance of these schemes under various network conditions.

2. RELATED WORK

A vast body of literature is devoted to video classification. Most work focus on the design of feature extraction and classification methods [7] [8], as well as the design of application-specific video data mining systems [3] [4] [5]. Prior research has also investigated operational rate-accuracy trade-off curves due to compression [9] [10], Our work stands apart from the above by tackling the key challenge of encoder adaptation in a networked video stream mining system, while accounting for dynamic and uncertain network conditions.

This work was partially support by NSF grant CNS 1016081. It is carried out while Cuiling Lan is visiting UCLA.

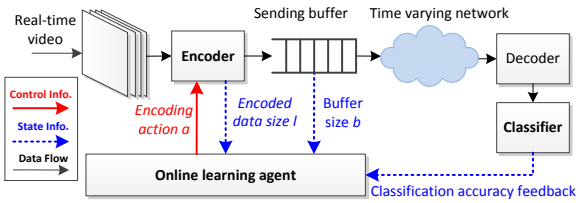

Fig. 1. System overview of networked video stream mining. The sender performs adaptive encoding based on observed sending buffer size, encoded data size, and classification accuracy feedback of previous segments from the receiver.

The technique of reinforcement learning has been widely explored in the context of wireless communications. Its applications range from call admission control in ATM networks [11] and interface switching in heterogeneous 4G networks [12] to energy-efficient wireless transmission [13] and congestion control for conversational services over wireless [14]. To the best of our knowledge, this paper is the first in applying reinforcement learning to networked video stream mining.

3. SYSTEM OVERVIEW

Figure 1 provides an architectural overview of the networked video stream mining system. At the sender, captured raw video streams are encoded on a segment by segment basis. The online learning agent dynamically determines the encoder configuration, such as quantization parameter (QP), for the upcoming segment, based on observed sending buffer size, encoded data size of the current segment, and classification accuracy feedback of previous video from the receiver. The encoded video streams are then temporarily cached at the sending buffer, and subsequently transmitted over the network. The receiver decodes the video and passes the extracted features to the classifier. Finally, classification results are compared with ground-truth class labels; classification accuracy information is reported back to the sender to assist its further learning.

In this work, we consider human action recognition as a concrete application example. The next section describes in detail the feature extraction process from motion vector (MV) information, as well as the multi-class support vector machine (SVM) classifier. Note that the overall system architecture applies more broadly to other types of features, classifiers, and mining applications.

4. FEATURE EXTRACTION AND SVM CLASSIFIER FOR HUMAN ACTION RECOGNITION

4.1. MV-Based Feature Extraction

It has been observed in [2] that motion information in the video provides important hints for recognizing human actions. Following an approach similar to [2], we extract features from motion vectors (MVs) available in the encoded bitstream. The main benefit is that features can be extracted directly as side products during the decoding process, thereby involving very low computational overhead.

The video segments are compressed using `x264` [15], a fast implementation of the H.264/AVC standard [16], with variable block sizes and a typical IPPP structure in each group of pictures (GOP). We record the decoded motion vectors for each 4×4 sub-block and normalize them based on reference frame distance [2]. For each video segment, the following set of features are extracted:

- *Histogram of one-dimensional (1D) MV features*: distribution of horizontal and vertical components of motion vectors. For each component, the signed motion values are quantified to 12 bins.
- *Histogram of two-dimensional (2D) polar features*: distribution of the direction and amplitude of motion vectors in polar coordinate system. The polar coordinate plane is partitioned into 8 different directions, with 6 amplitude levels per direction.
- *Motion History Image (MHI)*: a scalar-valued image which records the history of the spatial and temporal positions of motions for each 4×4 sub-block. The value of $\mathcal{H}_\tau(x,y,t)$ for each block is calculated as :

$$\mathcal{H}_\tau = \begin{cases} \tau, & \text{if} |MV_x| + |MV_y| > MV_{th} \\ \max(0, \mathcal{H}_\tau(x,y,t-1) - \delta), & \text{otherwise.} \end{cases} \quad (1)$$

  Here, x, y indicate the sub-block position in a frame; the frame index is t. The threshold value MV_{th} is chosen to identify significant motions while ignoring noise motions. The parameters τ and δ control the effective observation time window.
- *Motion Amplitude Image (MAI)*: a scalar-valued image recording the amplitude of motions over time and space for each 4×4 sub-block. Similar to MHI, each value of $\mathcal{M}_\tau(x,y,t)$ is calculated as:

$$\mathcal{M}_\tau = \begin{cases} 0.9\mathcal{M}_\tau(x,y,t-1) + 0.1(|MV_x| + |MV_y|), \\ \quad \text{if} |MV_x| + |MV_y| > MV_{th} \\ 0.9\mathcal{M}_\tau(x,y,t-1)), \text{ otherwise} \end{cases} \quad (2)$$

- *Number of blocks with large motion*: total count of sub-blocks containing large motion vectors ($|MV_x| + |MV_y| > MV_{th}$) in each video segment.

Altogether, there will be 393 extracted features per segment for the purpose of human action recognition.

4.2. Support Vector Machine (SVM) Classifier

We use the multi-class support vector machine (SVM) classifier from [17] to classify the human action videos with input features generated from motion vectors. The SVM technique is based on two key operations: mapping the input vectors to a high-dimensional feature space using some kernel function and constructing an optimal hyperplane for separating the transformed feature vectors [18].

In the first operation, the original feature vectors are mapped to a high-dimensional space and may become linearly separable. The mapping is typically carried out via kernel functions. Commonly used kernels include linear, polynomial, radial basis function (RBF), and sigmoid kernels. In this work, we have chosen the RBF kernel for its general applicability. Parameters of the RBF kernel are optimized via grid search over the parameter space, as proposed in [19].

Next, the classifier constructs a separating hyperplane to maximize the margin of separation between positive and negative training samples. Multi-class problems are broken down into multiple binary classification problems. We choose the default "one-against-one" strategy in the multi-class SVM classifier from [17], which assigns class labels using error-correcting codes based on multiple binary

classifier outputs [20]. To avoid over-fitting, the SVM classifier is trained with five-fold cross validation over two-thirds of the segments in the data set. The remaining ones are reserved for testing.

5. MDP-BASED PROBLEM FORMULATION

The problem of adaptive encoding at the sender can be formulated within the framework of Markov Decision Process (MDP). In the following, we describe the system states, actions, instantaneous rewards and overall returns of this formulation.

5.1. System States and Actions

We index the video segments by n, $n = 0, \cdots, \infty$, and designate the duration of each segment as T. Given a finite sending buffer limit B, the sending buffer size before encoding the n-th segment is $b_n \leq B$. The system state corresponds to the buffer size value in a discrete set $s_n \in \mathcal{S} = \{B_0, B_1, \cdots, B_K\}$:

$$s_n = \mathcal{Q}(b_n) = B_k, B_k \leq b < B_{k+1}. \tag{3}$$

Here, $B_0 = 0$, $B_K = B$,$\Delta B = B/K$, and $B_{k+1} = B_k + \Delta B$. The mapping function is represented by $\mathcal{Q}(\cdot)$.

The encoder action corresponds to the choice of quantization parameter (QP), and can be chosen from a set of candidate options: $a_n \in \mathcal{A}$. In our experiments, for instance, $\mathcal{A} = \{32, 35, 38, 41\}$ and $|\mathcal{A}| = 4$.

The encoded video data size for the n-th segment is l_n. The sending buffer size b_n evolves over time as:

$$b_{n+1} = \min[B, \max[0, b_n + l_n - c_n T]]. \tag{4}$$

where c_n indicates current network bandwidth.

5.2. State Transitions

The actual encoded data size l_n can be observed only *after* the encoding takes place. As Fig. 2 illustrates, the same action a may results in highly variable encoded data size l, depending on video content characteristics. We model the system state transition as a Markov chain. Its transitional probability can be derived as:

$$p_s(s_{n+1}|s_n, a_n) = p_b(b_{n+1}|b_n, a_n, c_n) = p_l(l_n|a_n)p_c(c_n). \tag{5}$$

Characterization of p_s requires knowledge of two independent processes: encoded data size at a given QP and evolution of network bandwidth over time.

5.3. Rewards and Returns

The instantaneous reward of the system comprises two components: buffer reward and classification reward.

$$r_n = \lambda(B - b_n - l_n) + (1 - \lambda)\mathbf{I}\{h_n = h_{n,o}\}. \tag{6}$$

Their relative weights are determined by the choice of λ. In (6), the first term indicates buffer reward, and corresponds to the remaining buffer budget after encoding the n-th segment. Note that the buffer size grows in proportion with sending delay of the encoded video segment. For the second term on classification reward, h_n denotes the classification label for the n-th segment and $h_{n,o}$ denotes its ground-truth label. The value of the indicator function $\mathbf{I}\{\cdot\}$ is one for correct classification (i.e., h_n matches $h_{n,o}$), and zero otherwise.

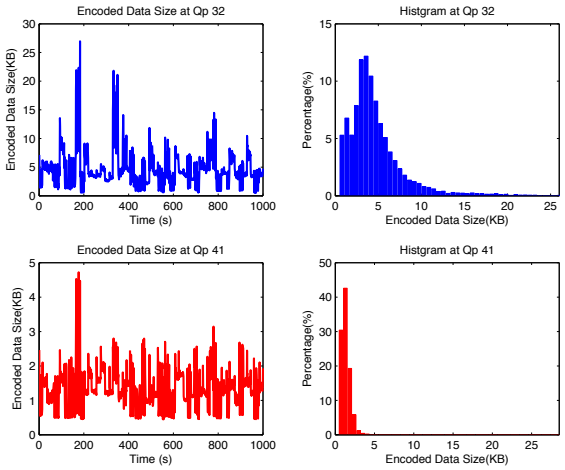

Fig. 2. Traces and distribution graphs of encoded video data size l_n at two different encoding actions: QP=32 and QP=41, respectively.

Consequently, overall return of the system is defined as:

$$R = \sum_{n=0}^{\infty} \gamma^n r_n, \tag{7}$$

with a discount factor of $0 \leq \gamma < 1$.

6. REINFORCEMENT LEARNING ALGORITHMS

Given (7), the goal is to find an optimal policy π^* for each state $s \in \mathcal{S}$, so as to maximize the expected return over all realizations of s_n:

$$\pi^* = \arg\max_{\pi: s \to a} \mathbf{E}R. \tag{8}$$

If the state transitional probabilities in (5) are known *a priori*, the optimal policy could be found via classic methods such as policy or value iteration [21]. In networked stream mining systems, however, both video content characteristics and network bandwidth may vary over time with unknown dynamics. In this paper, we propose to employ reinforcement learning algorithms to arrive at the optimal policy.

6.1. Myopic Optimization

A straight-forward approach is to keep record of past observations of instantaneous reward for each taken action a at system state s, in the form of a state-action value function $Q(s, a)$ [21]. However, the value of reward function as in (6) may not be available immediately after encoding the n-th segment. The classification accuracy reward may remain unknown since the encoded video segment may still be held at the sending buffer, or the classifier at the receiver has not yet finished processing previous segments. To resolve this issue, we propose to calculate an estimated reward instead, as:

$$\hat{r}_n = \lambda(B - b_n - l_n) + \begin{cases} 0, & b_n + l_n > B \\ (1 - \lambda)z_{n-k}, & \text{otherwise} \end{cases}. \tag{9}$$

In (9), the classification reward is set as zero if the sending buffer overflows after encoding the n-th segment. As an estimate, z_{n-k}

indicates the time-smoothed *previous* accuracy up to the $(n-k)$-th segments, with k outstanding segments either in the sending buffer or being processed at the receiver.

The value of the state-action function Q is updated as:

$$Q(s_n, a_n) \leftarrow (1-\alpha)Q(s_n, a_n) + \alpha \hat{r}_n. \quad (10)$$

The parameter α dictates the learning rate of the algorithm, i.e., how fast it adapts to changes in the environment.

This algorithm is myopic in nature, in that the Q function only accounts for the impact of the chosen action a_n at the current state s_n on the instantaneous estimated reward $\hat{r}_n$. It completely ignores the potential influence of a_n on system state transitions and future rewards.

6.2. Conventional Q-Learning

Alternatively, one may apply the conventional Q-Learning algorithm in [22] to solve for (8). In Q-learning, the sender also maintains a state-action function $Q(s, a)$, but updates it over time as:

$$\begin{aligned} Q(s_n, a_n) \quad \leftarrow \quad & (1-\alpha)Q(s_n, a_n) + \\ & \alpha[\hat{r}_n + \gamma \max_a Q(s_{n+1}, a)]. \end{aligned} \quad (11)$$

By adding the term of $\gamma \max_a Q(s_{n+1}, a)$, calculation of the Q values now accounts for the impact of a_n on expected future returns with discount factor γ. Note that setting $\gamma = 0$ reduces the Q-learning algorithm to myopic optimization.

6.3. Q-Learning with Virtual Experience (VE)

Although Q-learning can achieve optimal or near-optimal performance *after* convergence, it typically takes a long time to converge. The main reason for its slow convergence is that the learning agent can only update the value of Q for one combination of (s, a) at each step, and therefore needs to take many steps before it can fully explore a large state-action space.

In our problem, given the observed state transition from $s_n = \mathcal{Q}(b_n)$ to $s_{n+1} = \mathcal{Q}(b_{n+1})$ at a given QP, one may derive the rewards and state transitions at other systems states s'_n in a *deterministic* fashion, according to (4). Therefore, it is possible to simultaneously update $|S|$ entries in the state-action table according to (11), once the encoded data size l_n is obtained under chosen action a_n. This technique is called *virtual experience* update, and is shown in [13] to significantly improve learning speed.

Virtual experience update for other states $s'_n \neq s_n$ is as follows:

$$b'_{n+1} = \min[B, \max[0, s'_n + b_{n+1} - b_n]] \quad (12)$$

$$s'_{n+1} = \mathcal{Q}(b'_{n+1}) \quad (13)$$

$$\hat{r}'_n = \lambda(B - s'_n - l_n) + \begin{cases} 0, & s'_n + l_n > B \\ (1-\lambda)z_{n-k}, & \text{otherwise} \end{cases}. \quad (14)$$

Note that in (12), calculation of b'_{n+1} does not require explicit knowledge of the underlying network bandwidth c_n, but can be derived from observed buffer size changes $b_{n+1} - b_n$.

For all three schemes, we apply the ϵ-greedy method to balance between exploitation of learned knowledge and exploration of other actions. With probability ϵ, our scheme arbitrarily chooses the current action out of the entire action space $\mathcal{A}$; otherwise, it operates in greedy mode and chooses the action with the highest observed Q value: $a_n = \arg\max_a Q(s_n, a)$. In order to facilitate tracking of non-stationary environments, the value of ϵ is chosen to be constant.

7. SIMULATION RESULTS

7.1. Simulation Setup

We now evaluate performance of the proposed online learning schemes using the application example of human action recognition. The video contents are taken from the KTH dataset [6], and contain six types of actions: running, jogging, walking, boxing, hand-waving, and hand-clapping. The original sequences are broken into one-second segments, i.e., $T = 1$ second. There are a total of 11328 segments.

In our simulation, the sender loops over all segments for 15 rounds, with random ordering of the segments in each round, so as to mimic the scenario of long-running video stream mining. The video segments are encoded using x.264 [15] with IPPP GOP structure at 25 frames per second (fps). Encoded video segments temporarily stay in the sending buffer before being transmitted over the network. The sending buffer limit is set at $B = 80$ KB. The system states $s \in \mathcal{S}$ correspond to 32 discrete values of the sending buffer size, with equal increments of $\Delta B = 2.5$ KB.

At run time, the encoder has the option of encoding the upcoming video segment with a quantization parameter (QP) of 32, 35, 38, or 41, based on classification accuracy of previous segments and current sending buffer size. At the receiver, video segments are classified by the SVM classifier described in Section 4, which is trained with contents encoded at the highest quality (QP=32). The classification accuracy information is fed back to the sender, to assist its further learning of the encoding policy.

We consider three online learning algorithms, as described in Section 6: *myopic optimization*, *conventional Q-learning*, and *Q-learning with virtual experience (VE)*. In all three schemes, the sender takes the ϵ-greedy approach [21] to strike a balance between exploration and exploitation. The exploration ratio is set at $\epsilon = 0.1$: the encoder spends 10% of the time randomly selecting a QP, instead of abiding by the optimal action.

7.2. Transient Learning Behavior

We first consider a simple scenario with a fixed network bandwidth of $C = 30$ Kbps. The buffer size limit of 80 KB corresponds to a maximum sending delay of 21.3 seconds. Figure 3 shows traces of time-averaged overall reward and sending delay from all three schemes. It can be noted that both Q-learning schemes converge to higher overall rewards and lower sending delays than myopic optimization. While all the three schemes lead to similar classification accuracies around 78% (omitted in the figure), both Q-learning schemes achieve significantly lower buffering delays: around 2 seconds instead of 11 seconds as in myopic optimization. As expected, the proposed virtual experience technique helps to significantly accelerate convergence in Q-learning. Instead of spending over 80,000 seconds (about 22 hours) to converge to the optimal solution in conventional Q-learning, Q-learning with VE takes only about 2,000 seconds (about 33 minutes).

Figure 4 shows how various learning schemes react to changes in network bandwidth. When bandwidth drops abruptly from 40Kbps to 20Kbps, buffer delay increases for all schemes. Note, however, both Q-learning schemes settle at a much lower delay than that achieved by the myopic scheme. They also consistently maintain a higher overall reward over the myopic scheme. It can also be observed that the proposed Q-learning with VE can adapt to abrupt bandwidth change almost immediately, whereas Q-learning requires much longer time to converge to the right decisions. and cannot follow the change of bandwidth.

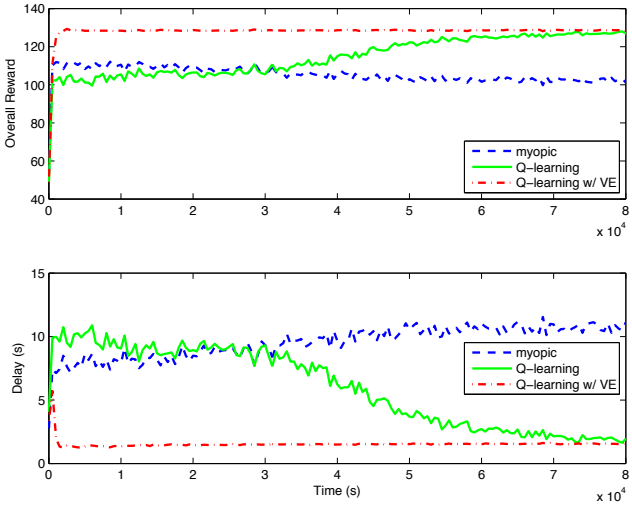

Fig. 3. Traces of overall reward and sending delay for myopic optimization, conventional Q-learning, and Q-learning with VE. The network bandwidth is fixed at $C = 30$ Kbps. The weighting factor in the reward function is chosen as $\lambda = 0.1$. Each data point in the trace corresponds to time-windowed average across 500 seconds, over 40 independent realizations.

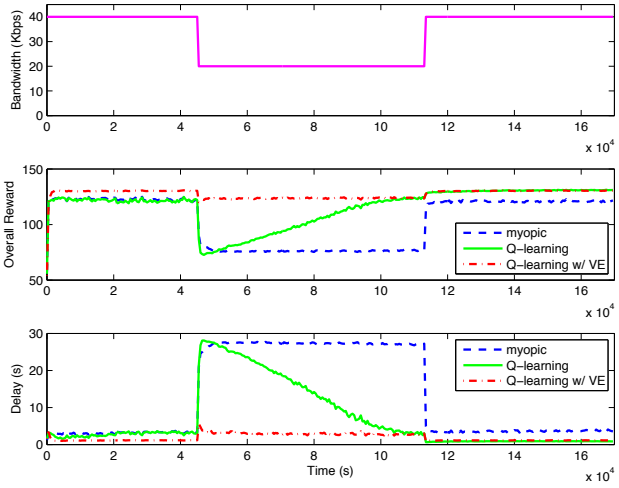

Fig. 4. Traces of network bandwidth, overall reward, and sending delay for myopic optimization, conventional Q-learning, and Q-learning with VE. The weighting factor in the reward function is chosen as $\lambda = 0.1$. Each data point in the trace corresponds to time-windowed average across 500 seconds, over 40 independent realizations.

7.3. Delay-Accuracy Tradeoff

Next, we consider the delay-accuracy tradeoff achieved by myopic optimization versus Q-learning with VE, by varying the weighting factor λ in the reward function. As Fig. 5 indicates, performances of both schemes are similar at two extreme ends of the tradeoff curve. When the importance of either lower buffer size (or higher classification accuracy) significantly outweighs the other component, either scheme eventually learns to encode all video segments at the highest QP (or lowest QP). For intermediate values of λ, Q-learning with VE can achieve a lower sending delay at higher classification accuracy over the myopic scheme.

7.4. Impact of Network Bandwidth

Finally, Fig. 6 shows overall reward and sending delay, as achieved by myopic optimization and Q-learning with VE, under various network conditions. All experiments are carried out with the same weighting factor $\lambda = 0.1$ in overall reward. As network bandwidth increases from 20 to 50 Kbps, Q-learning with VE consistently achieves a higher overall reward over myopic optimization. While sending delay from the myopic scheme ranges between 1.9 and 27.0 seconds, Q-learning with VE maintains a low delay between 0.8 and 2.5 seconds. Reduction in delay is more pronounced at lower network bandwidth: over 90% for $C = 20$ Kbps and over 85% for $C = 30$ Kbps.

8. CONCLUSIONS AND FUTURE WORK

This paper has explored the performance trade-off between sending delay and classification accuracy in bandwidth-constrained video stream mining systems. We have formalized the encoder adaptation problem as an infinite horizon Markov decision process (MDP), and have considered several low-complexity, model-free online learning solutions. We show that in comparison with myopic optimization, Q-learning scheme with virtual experience (VE) update yields significant delay reductions under various network conditions. Moreover, adoption of the VE technique drastically speeds up convergence from conventional Q-learning, allowing the encoder to adapt to abrupt changes in the environment on the order of minutes, instead of hours. While all performance evaluations have been carried out using human action recognition as a concrete example, our proposed MDP-based formulation and online learning algorithms are more generally applicable.

One of the limitations in this work is the coarse time granularity at which the encoder can adapt: once per second. This has restricted the achievable delay-accuracy performance trade-off in our current system. For future work, we plan to investigate how to perform online learning on a finer time scale, e.g., by adapting the encoder on a per-frame basis. It is also interesting to consider jointly optimizing encoder adaptation and classifier configuration under dynamic environments

9. REFERENCES

[1] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 3, pp. 257–267, Mar. 2001.

[2] R. Babu and K. Ramakrishnanb, "Recognition of human actions using motion history information extracted from the compressed video," *Image and Vision Computing*, vol. 22, no. 8, pp. 597–607, Aug. 2004.

[3] S. C. Chen, M. L. Shyu, C. Zhang, and J. Strickrott, "Multimedia data mining for traffic video sequences," in *Proc. 2nd International Workshop on Multimedia Data Mining, in conjunction with 7th ACM SIGKDD International Conference on*

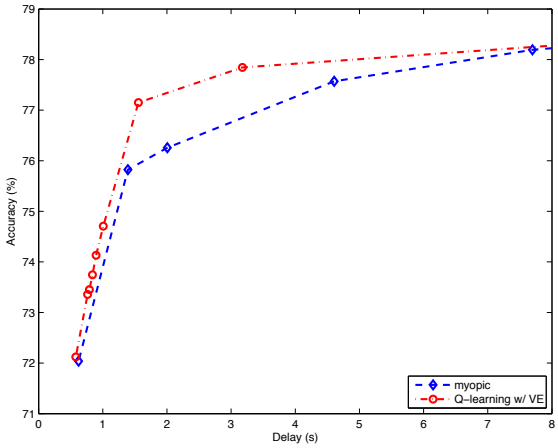

Fig. 5. Delay-accuracy tradeoff achieved by myopic optimization and Q-learning with VE, for various choices of the weighting factor λ in overall reward. The network bandwidth is fixed at $C = 30$ Kbps. Results are averaged across all video segments after convergence, over 40 independent realizations.

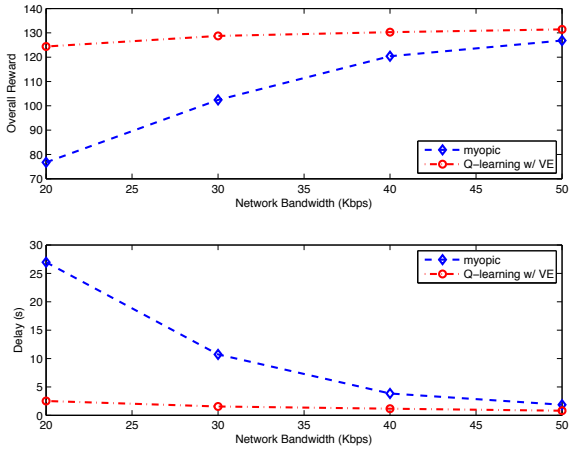

Fig. 6. Overall reward and sending delay, as achieved by myopic optimization and Q-learning with VE, at various network bandwidth values. The weight factor in overall reward is chosen as $\lambda = 0.1$. Results are averaged across all video segments after convergence, over 40 independent realizations.

Knowledge Discovery and Data Mining (MDM/KDD'01), San Francisco, CA, USA, Aug. 2001, pp. 78–86.

[4] J. Oh, J. Lee, and S. Kote, "Real time video data mining for surveillance video streams," in *Proc. 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'03)*, Seoul, Korea, Apr. 2003, pp. 222–233.

[5] X. Zhu, X. Wu, A. Elmagarmid, Z. Feng, and L. Wu, "Video data mining: semantic indexing and event detection from the association perspective," *IEEE Trans. Knowledge and Data Engineering*, vol. 17, no. 5, pp. 665–677, May 2005.

[6] "Recognition of human actions - KTH." [Online]. Available: http://www.nada.kth.se/cvap/actions

[7] M. Gaber, A. Zaslavsky, and S. Krishnaswamy, *A survey of classification methods in data streams*. Springer Verlag, 2007, vol. Chapter 3, pp. 39–59.

[8] D. Brezeale and D. J. Cook, "Automatic video classification: a survey of the literature," *Trans. Systems, Man, and Cybernetics (SMC), Part C*, vol. 38, no. 3, pp. 416–430, May 2008.

[9] B. M. Schmanske and M. Loew, "Effects of lossy wavelet-based compression on classification accuracy," in *Proc. IEEE International Geoscience and Remote Sensing Symposium (IGARSS'01)*, vol. 1, Sydney, Australia, July 2001, pp. 106–108.

[10] A. Z. Torres, R. Vitulli, and P. F. Xavier, "Impact of CCSDS-IDC and JPEG 2000 compression on image quality and classification," *Journal of Electrical and Computer Engineering*, vol. 2012, 2012.

[11] A. F. Atlasis, N. H. Loukas, and A. V. Vasilakos, "The use of learning algorithms in atm networks call admission control problem: a methodology," *Computer Networks*, vol. 34, no. 3, pp. 341–353, 2000.

[12] M. A. Khan, H. Tembine, and A. V. Vasilakos, "Game dynamics and cost of learning in heterogeneous 4G networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 1, pp. 198–213, Jan. 2012.

[13] N. Mastronarde and M. van der Schaar, "Fast reinforcement learning for energy-efcient wireless communication," *IEEE Trans. Signal Processing*, vol. 59, no. 12, pp. 6262–6266, Dec. 2011.

[14] O. Habachi, Y. Hu, M. van der Schaar, Y. Hayel, and F. Wu, "MOS-based congestion control for conversational services in wireless environments," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 7, pp. 1225–1236, Aug. 2012.

[15] "x264," http://developers.videolan.org/x264.html.

[16] *Advanced Video Coding for Generic Audiovisual services, ITU-T Recommendation H.264 - ISO/IEC 14496-10 (AVC)*, ITU-T and ISO/IEC JTC 1, 2003.

[17] C.-C. Chang and C.-J. Lin, "LIBSVM – a library for support vector machines." [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[18] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[19] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, Tech. Rep., 2003. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[20] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113–141, Sept. 2001.

[21] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. Cambridge, MA, USA: MIT Press, 1998.

[22] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge, UK, 1989.