Deep Reinforced Bitrate Ladders for Adaptive Video Streaming

Tianchi Huang¹, Rui-Xiao Zhang¹, Lifeng Sun¹,²,³*
¹Dept. of CS & Tech., ²BNRist, ³Key Laboratory of Pervasive Computing, Tsinghua University.

ABSTRACT
In the typical transcoding pipeline for adaptive video streaming, raw videos are pre-chunked and pre-encoded according to a set of resolution-bitrate or resolution-quality pairs on the server-side, where the pair is often named as bitrate ladder. Different from existing heuristics, we argue that a good bitrate ladder should be optimized by considering video content features, network capacity, and storage costs on the cloud. We propose DeepLadder, a per-chunk optimization scheme which adopts state-of-the-art deep reinforcement learning (DRL) method to optimize the bitrate ladder w.r.t the above concerns. Technically, DeepLadder selects the proper setting for each video resolution autoregressively. We use over 8,000 video chunks, measure over 1,000,000 perceptual video qualities, collect real-world network traces for more than 50 hours, and invent faithful virtual environments to help train DeepLadder efficiently. Across a series of comprehensive experiments on both Constant Bitrate (CBR) and Variable Bitrate (VBR)-encoded videos, we demonstrate significant improvements in average video quality, bandwidth utilization, and storage overhead in comparison to prior work as well as the ability to be deployed in the real-world transcoding framework.

CCS CONCEPTS
• Information systems → Multimedia streaming;

KEYWORDS
Adaptive Video Streaming, Bitrate Ladder.

ACM Reference Format:

1 INTRODUCTION
The Global Internet Phenomena Report COVID-19 Spotlight [36] reveals that almost 80% of internet traffic consists of video, gaming, and social content. Among them, the video streaming service covers 57.64%, and at least 51.43% of video streams are delivered by adaptive video streaming technologies [7]. In such technologies, on the server-side, video streams are pre-chunked and pre-encoded at various bitrates or quality levels. On the client-side, users often apply adaptive bitrate (ABR) algorithms to pick the proper bitrates for chunks to ensure the quality of experience (QoE). We often call the server-side encoding settings as bitrate ladder (§2.1).

The majority of existing bitrate ladder optimization schemes often use fixed encoding ladders (known as one-size-fits-all [10]), or consider either video contents [8] or network conditions [32] and solve the problem mathematically (§2.1). In this paper, we show that the bitrate ladder should be optimized by taking several factors from different perspectives into account, including video contents, network capacity, and storage costs on the cloud. However, the methods based on mathematical models fail to tackle the problem due to the high complexity and hardly identifies the underlying correlation between these factors (§2.2).

We present DeepLadder, a per-chunk neural transcoding system. Technically, we use a neural network (NN) to determine the proper setting for each resolution autoregressively. Moreover, considering that the sequential decisions require different exploration depth and practice intensity during the training [22], we leverage a novel deep reinforcement learning (DRL) algorithm to balance a variety of goals such as maximizing perceptual video quality, improving bandwidth utilization, and reducing the storage overhead. In particular, we integrate the weight representation, the importance of the storage cost, into the input. The weight is randomly adjusted for each episode. Therefore, such many-goal technique [43] enables DeepLadder to handle the multi-objective reward function with dynamic storage weights (§3).

We design a faithful offline simulator to allow DeepLadder to converge within an acceptable time (§4.1). Besides, considering that DeepLadder requires a large corpus of videos to converge, we collect a video dataset with two encoding types (i.e., constant bitrate (CBR) and variable bitrate (VBR)-encoded videos). Using trace-driven experiments, we show that: i) Compared with existing methods under various networks and videos, DeepLadder-CBR improves the average quality by 8.49%-14.25%, enhances the bandwidth utilization by 3.66%-11.68%, and reduces the storage cost by 39.77%-54.84%. ii) Results on DeepLadder-VBR also illustrate significant improvements in the average quality (3.8%-7.7%) and reduction in terms of the storage costs (21.26%-44.88%). iii) DeepLadder is practical and can be deployed into the transcoding framework. Testing results on both CPU and GPU devices indicate that DeepLadder achieves the aforementioned performance with only 3.4% on extra costs compared with the original system while saving almost 70% of overall overhead in comparison to state-of-the-art heuristics (§4).

In general, we summarize the contributions as follows: i) We investigate three critical features for constructing a good bitrate ladder. Further, we propose DeepLadder, the first DRL-based approach to solve the problem. ii) We train DeepLadder over various video dataset and network traces. The bitrate ladders selected by DeepLadder demonstrate the superiority of the algorithm compared with prior approaches.
<table>
<thead>
<tr>
<th>Scheme</th>
<th>Bitrate Ladder (Kbps)</th>
<th>Storage (MB)</th>
<th>Quality (VMAF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDCHAS [6]</td>
<td>[43,100, 150, 200, 250, 300, 400, 500, 600, 700, 900, 1200, 1500, 2000, 2100, 2400, 2900, 3300, 3600, 4000]</td>
<td>13.5725</td>
<td>46.91</td>
</tr>
<tr>
<td>YouTube [2]</td>
<td>[90, 500, 700, 1250, 3750, 6000]</td>
<td>6.145</td>
<td>48.93</td>
</tr>
<tr>
<td>Pensieve [24]</td>
<td>[300, 750, 1200, 1850, 2850, 4300]</td>
<td>5.625</td>
<td>53.04</td>
</tr>
<tr>
<td>BitMovin [46]</td>
<td>[400, 800, 1200, 2400, 4800]</td>
<td>4.8</td>
<td>53.45</td>
</tr>
</tbody>
</table>

2 BACKGROUND AND MOTIVATION

In this section, we start by illustrating the background of the bitrate ladder. Then we list several challenges in terms of constructing a bitrate ladder. Finally, we describe how to use customized deep reinforcement learning methods to tackle the problem.

2.1 Related Work

Most Internet video sharing platforms adopt adaptive bitrate (ABR) technologies, e.g., DASH [13] and HTTP Adaptive Streaming (HLS) [1], to ensure quality of experience (QoE). In such technologies, raw videos are required to be pre-processed before being played on the client. Specifically, for each raw video, first, the video is chunked as a segment duration (e.g., 4 seconds). Then the segment chunks are encoded as different bitrates or quality levels. Finally, all the segments are stored in the assigned storage server. Each video has been previously pre-chunked and pre-encoded into different settings, i.e., resolution-bitrate for CBR-coded and resolution-quality for VBR-coded videos [26]. The Video transcoding problem has already been published for about two decades [5]. With the population of adaptive video streaming technologies [9], recent years have seen a number of studies on optimizing transcoding strategies. Such schemes are mainly organized into two types: mathematical-based and learning-based.

Mathematical-based. Modern bitrate ladder optimization method is started by Netflix [10]. In detail, De Cook et al. analyze all the possible resolution-bitrate pairs independently, and pick the best pair that is close to the convex hull as possible. Meanwhile, Chen et al. [8] propose a per-chunk encoding scheme that leverages player feedback behavior for optimizing the chunked transcoding pipeline. Moreover, Reznik et al. [32] first take the current network status into consideration and model the problem as a non-linear constrained optimization problem. Other studies [31] use two subsets of codecs, i.e., H.264 and H.265, to construct the multi-codec adaptive bitrate streaming system. Besides, Reznik et al. [30] discuss the optimization problem with different decoding on devices, video player playback features, as well as network capabilities. Meanwhile, some related work have already addressed the storage constraints [35, 41]. Technically, all of the schemes consider the problem as the non-linear optimization problem and solve it mathematically.

Learning-based. Recently several schemes have also been proposed to optimize the bitrate ladder via deep learning method. Kat-senou et al. [18] propose a content-gnostic approach which adopts machine learning technologies to estimate the bitrate ranges for different resolutions. Xing et al. [49] employ NN to predict the optimal rate-control target for User-Generated Content videos. However, such methods are often myopic, one-shot, and only consider instant rewards.

2.2 Motivation

To better investigate how to generate an outstanding bitrate ladder, we set up several representative experiments from different aspects, including contents, networks, and costs.

> Video Content Video content usually has a great impact on encoded video quality. To prove it, we collect three videos (No.1 - No.3) with different contents and report the Rate–distortion curve (R–D curve) [50] with the video resolution of 240p and 720p. Here we apply Video Multi-Method Assessment Fusion (VMAF) [28], the state-of-the-art video quality assessment, ranging from 0 to 100, as the video quality metric. We find the video content is quite diverse, since i) In the resolution of 240p (Figure 1(a)), the video resolution is too small, causing premature saturation for video No.3; ii) In 720p (Figure 1(b)), the video qualities can converge into an acceptable range. Moreover, Figure 1(c) shows the performance of a sequence of video chunks of the same video (games, H.264, CBR-encoded by FFmpeg [12] with the constant bitrates). We find that the video quality will noticeably change due to dynamic video contents. Figure 1(d) gives an example of video sequence encoded by VBR (music video, H.265, VBR-encoded with –crf=25), where the dashed lines mark the video sizes of four tracks. Results demonstrate the same conclusion that is notated in the CBR-encoded experiment. Thus, we argue that the typical one-fits-all strategy can hardly always achieve satisfactory performance for various video contents.

> Network Traffic Distributions. We further investigate the relationship between the average bandwidth distribution and the default bitrate ladder setting proposed by Mao et al. [24]. We consider...
Transcoding
Stage

Video Quality

Storage Cost

Video Chunks

Deploy

Online Stage

Server

Figure 3: An Overview of DeepLadder’s System Workflow.

3 DEEPLADDER SYSTEM MECHANISM

Big Picture. As illustrated in Figure 3, DeepLadder’s basic system workflow is mainly composed of the transcoding stage and the online stage. In the transcoding stage, we leverage a NN-based decision model for constructing the proper bitrate ladders and use the assigned settings to transcode the video for each chunk. In the online stage, the encoded chunks are deployed on the server and wait for fetching. The NN continually obtains the video quality and storage at the transcoding stage, as well as periodically receives the network status feedback at the online stage.

Why Sequential Learning? Recall that previous work solves the problem within one-step (§2.1). However, it’s impractical to directly apply the conventional DRL framework to solve the bitrate construction process since the action space will be extremely huge if the NN samples all the bitrates. Given the number of the bitrate candidates \( m \) and the count of the bitrate ladder \( n \), for \( s \) states, each iteration needs \( O(m^s n^2) \) steps, or slightly faster if there is sparsity in the transition function [39]. To eliminate the huge complexity, considering the lifetime of a bitrate decision as a Markov Decision Process (MDP), we pick the proper bitrate or quality level for each resolution in an autoregressive manner. Such sequential learning settings enable the algorithm to effectively reduce the complexity of each iteration to \( O(mn^2) \).

3.1 NN Overview

DeepLadder’s NN architecture is illustrated in Figure 4. We now explain the states, actions, reward definition.

State. We mainly categorize the DeepLadder’s input into four parts, that is video features, network capacity, past actions, and storage weights. In other words, for each video chunk \( t \), we have the state-space \( S_t = [F_t, N_t, P_t, w] \).

» Video Frames \( F_t \). Recent research demonstrates that the perceived video quality heavily depends on the underlying attribute of video content. Thus, the DeepLadder’s learning agent takes the video frame \( F_t(t = 1, 2, \ldots, T) \) as the input, where the frame is the I-frame (intra picture) [20] of the given video. As suggested by recent work [4, 24], the videos are chunked as 4 seconds, i.e., \( T=4 \). We down-sample each of the raw video frames to size \( 224 \times 224 \times 3 \), feed it into the pre-trained ResNet50 [16] model, and get output feature maps from the last convolution layer. A global average pooling (GAP) layer is applied to merge the information from all feature maps. We call the process as video extraction (§4.4).

» Network Capacity \( N_t \). Recall that the current global network traffic distribution is a non-trivial feature for optimizing the ladder. Furthermore, How to accurately represent the network state is quite challenging since all the bandwidth traces can be viewed as a continuous sequence instead of a finite discrete vector. Hence, we apply the bandwidth trace histograms with average throughput for representing current network capacity. Specifically, we categorized the average bandwidth within a specified range of values \( n \) (called bins) and static the frequency of the data values. We set \( n=20 \) to balance the performance and the cost.

» Past Actions \( P_t \). The agent takes past actions’ sequence \( P_t = [a_0, \ldots, a_{t-1}] \) into NN, where \( a_i \) reflects the normalized action for video resolution \( i \). Please note that the action representation is different in the CBR and VBR coding (§4).

» Storage Weights \( w \). We consider the diverse settings of storage weight for each content provider. For instance, some providers prefer video quality and consider the storage cost as a non-trivial metric, or vice versa. To overcome the fact that the NN will be retrained if the required storage weight changes, we take the storage weight \( w \) into the state. We randomly sample weights \( w \in (0, 1] \).
for each episode, enabling the NN to learn the correlation between the storage weight and the feedback reward [45].

**Action.** Considering that DeepLadder is able to support the diversity of coding methods, we will introduce the action space for DeepLadder-CBR and DeepLadder-VBR in §4.

**Reward.** We list the instant reward function $r_t$ in Eq. 1. Given a network condition $C$, where $C$ is often represented as a list of saturated network traces [47], we aim to maximize the bandwidth utilization and average video quality, and in turn, minimizing the storage cost. Detailing the equation, that includes:

- **Bandwidth Utilization $U$:** represents the actual network utilization of the selected chunk size in the current network state (Eq. 2), where $Br(a_t)$ denotes the bitrate for the picked chunks, $C_{avg}$ means the network bandwidth under all actions $a_t$. Here $p(a_t|C)$ means the probability that the chunk $a_t$ being selected over the given network condition $C$. Note that the result of $p(a_t|C)$ is highly correlated with the ABR algorithms. Considering the diversity of ABR algorithms in the online stage, we refer the default ABR algorithm as the oracle policy. The reason is that end-to-end optimization is impractical in practice since bitrate construction and ABR algorithms will be optimized by different methods and people (or groups), while the main goal for that two algorithms is to achieve ABR oracle as much as possible.

- **Average video quality $Q$:** computed by the expectation of the video quality $Q(a_t)$ being selected by the action sequence $[a_0, \ldots, a_t]$ for chunk $[0, \ldots, t]$.

- **Storage Cost $S$:** means the average chunk size for the action sequence $[a_0, \ldots, a_t]$. Recall that the storage weight $w$ is meant to balance the positive and negative metrics.

$$r_t = \sum_t p(a_t|C)U(a_t, C) + \sum_t p(a_t|C)Q(a_t) − w \sum_t S(z(at))/t \cdot (1)$$

$$U(a_t, C) = \begin{cases} Br(a_t)/C_{avg} & \text{if } S(z(a_t)) \leq C_{avg} \\ 1 − Br(a_t)/C_{avg} & \text{if } S(z(a_t)) > C_{avg} \end{cases} \quad (2)$$

**3.2 NN Architecture.**

Now we introduce DeepLadder’s NN architecture, which is composed of video extraction process and the inference process. In the video extraction process, for the content-aware video frame, we employ the state-of-the-art pre-trained image classification model to extract spatial information from the video frames. At first, we feed the I-frames of the video into the pre-trained ResNet50 [16] model and output feature maps from its last convolution layer. We then apply a global average pooling (GAP) layer to merge the information from each feature map. Next, the features are down-sampled by an 1D-CNN layer with kernel size=4, filter number=128, and stride=1. So we take the video frames sized [4, 224, 224, 3] as the input. The video extraction network outputs the vector with the size of [4, 128]. In the inference process, for the network state representation, we adopt a 1D-CNN-layer with kernel size=2 and filter number=128 to over-sample the features to a 128-dim vector. What’s more, we use two 128-dim fully connected layers to extract the features from past actions and storage weights respectively. Then all the vectors are integrated into a concatenate layer. Finally, the outputs of the DeepLadder are the policy network and the value network, in which the activation function of the policy network is softmax and we set linear function for value network. In this work, we use Dual-clip PPO [51] to jointly optimize the policy for both long-term reward and entropy.

**4 EVALUATION**

In this paper, we attempt to evaluate DeepLadder on both CBR-encoded and VBR-encoded videos. The main challenge faced by many experiments is the CBR and VBR are two different coding methods [26]: CBR controls the transcoding process by adjusting the bitrate, as VBR controls the video quality. The design principle of the action space of two schemes is described in §4.2 and §4.3.

**4.1 Methodology**

The **Video Dataset.** Deep learning requires a large amount of training data due to the huge number of parameters to be tuned. Nevertheless, revisiting previously proposed public video datasets [21, 27, 53], we observe that the existing datasets lack either the diversity of video contents or the video coding types, which are unable to be used for research purpose directly. We make two contributions:

- For the CBR (constant bitrate) coding, we collect a video dataset that contains 87 H.264 videos, 9 tracks/levels, 4714 video clips. Moreover, we measure the VMAF [28] metric for each video chunk with different levels and resolutions, yielding 254,529 samples.

- We make a VBR (variable bitrate)-encoded video dataset, which includes 65 H.265 videos, 20 tracks/levels, totaling 3509 video clips. Meanwhile, we apply SSIM [45] and PSNR [17] metric to measure the video quality for each video chunk and get 842,160 samples.

The **Network Dataset.** To better simulate the diversity of real-world network conditions, we collect over 3,000 network traces with a duration of almost 50 hours, from public datasets for training and testing. The dataset contains traces from HSDPA [33], FCC [29], Oboe [42], and HDFS [38] datasets. In general, for videos, we randomly divide the dataset into two parts, 80% traces for training and 20% for testing. For network traces, we train the NN on the training set and test the performance on other network datasets.

**Implementation.** We adopt TensorFlow [3] to implement the DeepLadder’s training workflow and apply TFLearn [40] to implement the NN architecture. Moreover, we use the pre-trained ResNet-50 model in Keras [15] to extract features of video contents. We set the learning rate of the policy network as $10^{-5}$ and set that of the value network as $10^{-4}$, and leverage the Adam optimizer [19] to optimize the NN. Consistent with standard YouTube settings [2], the algorithm will terminate after generating 6 bitrates. At each
Table 2: Comparing the execution time and accuracy of the DeepLadder’s simulator with transcoding in the real-world.

<table>
<thead>
<tr>
<th>Simulator (ms)</th>
<th>In situ (ms)</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Execution</td>
<td>Trans</td>
</tr>
<tr>
<td>CBR</td>
<td>0.018</td>
<td>2010</td>
</tr>
<tr>
<td>VBR</td>
<td>0.0028</td>
<td>2186</td>
</tr>
</tbody>
</table>

Figure 5: Comparing DeepLadder-CBR with existing bitrate optimization approaches over the HSDPA and FCC traces. Results are collected based on different videos and network traces respectively. We set \( w = 0 \).

step, the learner picks the proper profile for the video in the resolution of \{144p, 240p, 360p, 480p, 720p, 1080p\} respectively. We train DeepLadder’s NN decision core on the laptop with Intel i7 3.7GHZ, 12-core CPUs, 64G RAM, and a Nvidia-1080TI GPU. We use multi-agent training technologies that enable: 16 forward propagation learning agents, placed on the CPU, for sampling; 1 central agent, placed on the GPU side, for training. Training takes approximately 12-16 hours. Recall that the cost is incurred in the offline stage, as we deploy the learned policy on the online stage straightforwardly for zero-shot inference [44].

Transcoding Simulator for Fast Training. Considering that the learning agent may repeatedly take the same bitrate ladder configuration, we pre-transcode every decision of each video and notes the features extracted by ResNet-50 [16], the video qualities (i.e., PSNR, SSIM, VMAF), as well as video sizes after transcoding. In particular, considering the continuous action space in the CBR-encoded video environments, we estimate the video quality and the video size via the piece-wise linear-regression method. Table 2 shows the proposed simulator performs at least 15,000× acceleration with an accuracy of 98.83%.

Baseline. We choose several representational bitrate ladder optimization algorithms from both academia and industry (§2.1). For the CBR-encoded video scenario, we set:

- **Netflix [10]** (NF-Greedy): offline picking the results of all the bitrate-resolutions pairs for each resolution, and select the best pair which is close to the convex hull as possible. We add the algorithm into the baseline since it’s scalable, simple yet effective.

- **Pensieve-Settings [24]** (Pen-Set): a popular fixed encoding ladder profile, which encodes the videos at the bitrate of \{300, 750, 1200, 1850, 2850, 4300\}kbps.

- **MPC-Settings [52]** (RMPC-Set): another fixed encoding ladder settings with \{300, 700, 1000, 1500, 2000, 3000\}kbps.

Furthermore, we select the following baselines for the VBR-encoded video scenario:

- **Fixed-25 [26]**: recent studies find that using the Constant Rate Factor (CRF) value of 25 can provide good viewing quality [11].

- **X265 [48]**: the default CRF value is fixed as 28 in libx265 [34].

4.2 DeepLadder-CBR

In this section, we analyze the performance of DeepLadder with constant bitrate (CBR)-encoded videos. Specifically, we use VMAF [28], a state-of-the-art objective full-reference perceptual video quality metric, to measure the video quality.

Action Space Design. Considering the high complexity of continuous action selection of CBR-encoded videos, we downscale the continuous action space to the discrete one. In detail, we set the the action \( a_{i} = N_{i} \), where \( N_{i} \) represents the i-th throughput categorized by the bandwidth histogram method (§3.1). Meanwhile, we apply masked-softmax to mask previous selected actions. In conclusion, we have 20 actions in total, consistent with the number of network status \( n \) of the NN’s state.

DeepLadder-CBR vs. Baselines. Figure 5 shows the comparison between DeepLadder with recent proposed schemes (§4.1). First, as shown in Figure 5(a) and Figure 5(c), we test the performance of each scheme by the video under the HSDPA and FCC network environment and report the average video quality, bandwidth utilization, and storage overhead. Here error bars span ± one standard deviation (std) from the average. We can see that DeepLadder outperforms previously proposed methods, with the improvements on i) average video quality of 8.49% - 12.94% on the HSDPA dataset and 8.78%-14.25% on the FCC dataset; ii) 3.66%-11.68% in terms of the bandwidth utilization on HSDPA. Meanwhile, DeepLadder reduces the storage overhead by 39.77%-54.84% compared with the baselines. In particular, despite the outstanding results that the fixed encoding ladder achieves, both RMPC-Set and Pen-Set consume too much storage overhead compared with DeepLadder. What’s more, we treat the network trace in the dataset as an independent network condition, aiming to evaluate the performance of the proposed schemes against others in a fine-grained perspective. Figure 5(b) and Figure 5(d) demonstrate that DeepLadder is well behaved on the average video quality, which is always better than other baselines.

Comparison of Different Storage Weights \( w \). Recall that we take the storage weight as a goal into the DeepLadder’s NN input, so there’s no need to be retrained when adjusting the storage weight as needed. Figure 6 illustrates the curve of average quality...
Figure 7: Comparing DeepLadder-VBR with existing bitrate optimization approaches on the Oboe and HDFS network traces.

Figure 8: Training DeepLadder-VBR with 30% and 100% of videos.

Table 3: The total timespan comparison of DeepLadder and baselines in the real-world deployment.

<table>
<thead>
<tr>
<th>Process (ms)</th>
<th>Video Extraction</th>
<th>Inference Timespan</th>
<th>Trans. Time</th>
<th>Extra Cost (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepLadder@CPU</td>
<td>439</td>
<td>3</td>
<td>12060</td>
<td>3.67</td>
</tr>
<tr>
<td>Netflix@CPU</td>
<td>-</td>
<td>-</td>
<td>39937</td>
<td>332.0</td>
</tr>
<tr>
<td>DeepLadder@GPU</td>
<td>94</td>
<td>12</td>
<td>3407</td>
<td>3.11</td>
</tr>
<tr>
<td>Netflix@GPU</td>
<td>-</td>
<td>-</td>
<td>11849</td>
<td>347.8</td>
</tr>
</tbody>
</table>

4.3 DeepLadder-VBR

We set up a new VBR-encoding testbed with the VBR-encoded video dataset, the VBR transcoding simulator, as well as network traces. To better understand DeepLadder’s generalization ability, we apply another classical video quality assessment $VQA^{log}$ [37, 54], which is the non-linear relationship between SSIM and mean opinion score (MOS), to measure the quality metric. Here we follow the recent work [54] to set the parameters (Eq. (3)), in which $SSIM_t$ means the average SSIM metric for the video chunk $t$. Moreover, we utilize Oboe [4] and HDFS [38] network traces to validate the performance of the proposed methods.

$$VQA^{log}_t = 75.5 - \frac{65.4}{1 + e^{37.37 \times (SSIM - 0.93)}} + 24.4\cdot SSIM_t$$ (3)

Action Space Design. We set the action size of DeepLadder-VBR as 20, symbolizing that the videos will be encoded by $crf=20$-$39$. Such settings have covered the best quality for most videos [26].

DeepLadder-VBR vs. Existing Methods. Figure 7 illustrates the results for the Oboe [4] and HDFS [38] network conditions. As expected, DeepLadder-VBR ($w = 0$)’s bandwidth utilization is within 8.37%-15.03% compared with fixed encoding ladder schemes. Moreover, DeepLadder-VBR ($w = 1$) effectively preserves the average video quality $VQA^{log}$ (even improves the quality by 3.8%-7.77% compared with fixed encoding ladder approaches), and reduces storage cost by 29.8% on Oboe, 21.26%-44.88% on HDFS.

Training DeepLadder-VBR with More Data. The traditional bitrate ladder optimization scheme uses less video to get better results. So what about DeepLadder? To answer this question, we record the performance of the proposed scheme on the HSDPA network condition every 500 epochs. Results are reported with the training curve of two DeepLadder-VBRs in Figure 8, where one of them learns the policy by only 30% videos. Unsurprisingly, DeepLadder (100%) outperforms DeepLadder (30%) on the average reward of 3.0%. Such observation indicates that deep learning indeed requires a large number of data (i.e. our video dataset §4.1) for generalizing a good strategy.

4.4 Practical Implementation

It’s notable that the increased overhead will neglect the increased overall performance if it is significant. To this end, we deploy the proposed scheme as a service into a simple transcoding framework and list the time consumption for each process in Table 3, tested on an Intel Core i7 CPU 2.2GHz and an Nvidia 1080Ti GPU. In particular, in the GPU test, we use the FFmpeg’s integration of NVIDIA Video Codec SDK that enables high-performance hardware-accelerated video pipelines [25]. Results indicate that DeepLadder (in gray) only consumes 3.11%-3.67% on the extra cost compared with the original system. More surprisingly, we find that DeepLadder’s inference time (without video feature extraction) on GPU is 4x higher than that on CPU. It makes sense since the NN adopts a lightweight design so that it can achieve higher efficiency on the CPU device. Moreover, comparing with state-of-the-art heuristic Netflix [10], DeepLadder only requires 29.64% (GPU) and 31.3% (CPU) of overall overhead as well as obtains better performance ($\S$4.2). In general, we believe that DeepLadder is quite suitable and practical to be an add-on for real-world implementation, e.g., Morph [14].

5 CONCLUSION

DeepLadder is a novel learning-based bitrate ladder construction system. Unlike previous schemes, DeepLadder’s NN takes current raw video contents, network traffic capacities, and the storage overhead into the input. We adopt the DRL method to train DeepLadder w.r.t a large corpus of internet network traces and collected videos. Experimental results on both CBR and VBR-encoded videos indicate the superiority of DeepLadder over existing approaches, which sheds some light on optimizing the transcoding pipeline in a smart and practical manner.

Acknowledgments. We thank our shepherd Christian Timmerer, and the anonymous NOSSDAV reviewers for their constructive feedback. This work was supported by NSFC under Grant 61936011, 61521002, Beijing Key Lab of Networked Multimedia, and National Key R&D Program of China (No. 2018YFB1003703).