Deep Reinforced Bitrate Ladders for Adaptive Video Streaming

Tianchi Huang[1], Rui-Xiao Zhang[1], Lifeng Sun[1,2,3*]
[1]Dept. of CS & Tech., [2]BNRist, [3]Key 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:
Tianchi Huang, Rui-Xiao Zhang, Lifeng Sun. 2021. Deep Reinforced Bitrate Ladders for Adaptive Video Streaming. In *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '21)) (NOSSDAV '21), September 28-October 1, 2021, Istanbul, Turkey*. ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.1145/3458306.3458873

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 storage weight, representing 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 tp 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)-encoded 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).

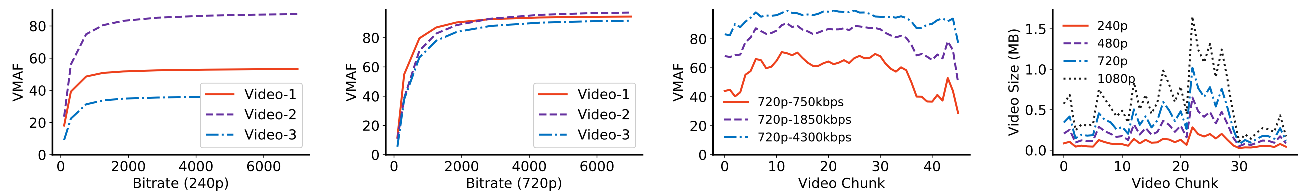
In general, we summarize the contributions as follows:

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.

NOSSDAV '21, September 28-October 1, 2021, Istanbul, Turkey

ACM ISBN 978-1-4503-8435-3/21/09...$15.00
https://doi.org/10.1145/3458306.3458873



(a) R-D curves: 240p (b) R-D curves: 720p (c) Quality: CBR video. (d) Size: VBR video.

Figure 1: Visualizing the correlations of video bitrates, perceptual video qualities, and the video contents.

Table 1: The typical bitrate ladder

Scheme	Bitrate Ladder (Kbps)	Storage (MB)	Quality ([28])
FDCHAS [6]	[45,100, 150, 200, 250, 300, 400, 500, 600, 700, 900, 1200, 1500, 2000, 2100, 2400, 2900, 3300, 3600, 4000]	13.5725	46.91
YouTube [2]	[90, 500, 700, 1250, 3750, 6000]	6.145	48.93
Pensieve [24]	[300, 750, 1200, 1850, 2850, 4300]	5.625	53.04
BitMovin [46]	[400, 800, 1200, 2400, 4800]	4.8	53.45

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. Katsenou 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

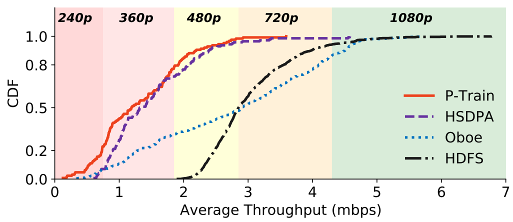


Figure 2: Cumulative distribution function (CDF) of session throughput on different network traces.

a set of the network trace datasets: Pensieve training dataset (P-Train) [23], HSDPA [33], Oboe [42], and HDFS [38] network traces. The CDF of session throughput curve in Figure 2 shows that the current network environments seldom cover all the range of bitrate candidates. For example, the average throughput of the trace in the P-Train dataset never reaches 4.3mbps, i.e., the bitrate setting of 1080p videos, while the average bandwidth of the trace in the HDFS dataset seldom works lower than 1.85mbps (the bitrate setting of 480p videos). Thus, motivated by the previous bitrate ladder scheme [31], we take the current network traffic conditions into consideration.

▹ **Storage Costs** Table 1 lists the typical bitrate ladder settings. Specifically, we compute the average storage consumed for each raw video chunk and use RobustMPC [52] to evaluate the performance of each bitrate ladders via VMAF [28] tools over the HSDPA [33] networks. Surprisingly, we observe that the bitrate ladder with too many tracks fails to obtain higher video qualities. What's more, BitMovin's bitrate ladder settings achieve optimal performance using 35.3% of the storage cost compared with FDCHAS[6]. So how to construct the bitrate ladder with lower storage costs?

Summary. In general, we find that constructing a good bitrate ladder should consider video content features, current network traffic capacities, and especially, the overall storage cost on demand. It's quite challenging for conventional heuristics to effectively combine such metrics from various perspectives. In contrast, we treat the bitrate ladder problem as a sequential decision-making process (§3) that fits well with the objectives of reinforcement learning (§4). Moreover, we leverage deep reinforcement learning (DRL) since its capabilities to *generalize* from raw fresh data without relying on handcraft engineering (§3.1).

3 DEEPLADDER SYSTEM MECHANISM

Big Picture. As illustrated in Figure 3, DeepLadder's basic system work-flow 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

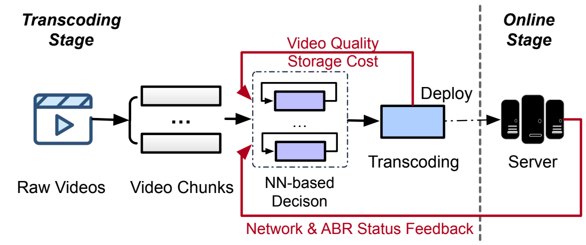


Figure 3: An Overview of DeepLadder's System Workflow.

candidates m and the count of the bitrate ladder n, for s states, each iteration needs $O(m^n s^2)$ 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 $O(mns^2)$.

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 $F_t(t = 1, 2, \dots, T)$ 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 *224×224×3*, 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 $P_t = \{a_0, \dots, a_{t-1}\}$ 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 $w \in (0, 1]$

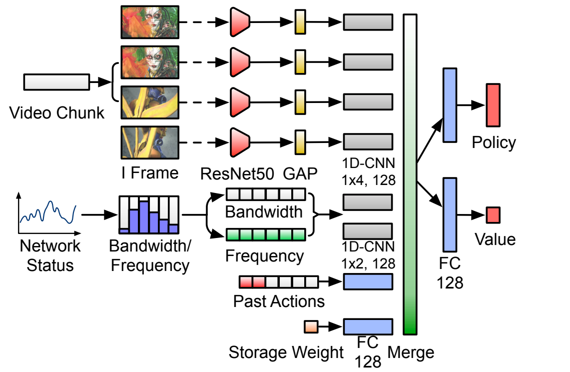


Figure 4: DeepLadder's NN Architecture Overview.

for each episode, enabling the NN to *learn* the correlation between the storage weight and the feedback reward [43].

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, $C_{a=at}$ 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 $\{a0, \ldots, a_t\}$ for chunk $\{0, \ldots, t\}$.

▹ ***Storage Cost*** Sz***:*** means the average chunk size for the action sequence $\{a0, \ldots, a_t\}$. Recall that the storage weight w is meant to balance the positive and negative metrics.

$$r_t = \underbrace{\sum_t p(a_t|C)U(a_t, C)}_{\textit{Bandwidth utilization.}} + \underbrace{\sum_t p(a_t|C)Q(a_t)}_{\textit{Video quality}} - w \underbrace{\sum_t Sz(at)/t}_{\textit{Storage cost}}. \quad (1)$$

$$U(a_t, C) = \begin{cases} Br(a_t)/C_{a=a_t} & Sz(a_t) \leq C_{a=a_t} \\ 1 - Br(a_t)/C_{a=a_t} & Sz(a_t) > C_{a=a_t} \end{cases} \quad (2)$$

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.

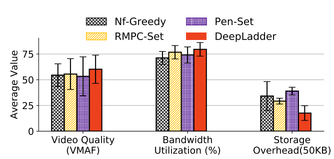
▹ We make a VBR (variable bitrate)-encoded video dataset, which includes 65 H.265 videos, 20 tracks/levels, totaling 3509 video clips. Meanwhile, we apply SSIM [45] and PSNR [17] metric to measure the video quality for each video chunk and get 842,160 samples.

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.

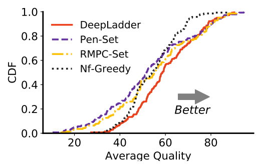
Implementation. We adopt TensorFlow [3] to implement the DeepLadder's training workflow and apply TFlearn [40] to implement the NN architecture. Moreover, we use the pre-trained ResNet-50 model in Keras [15] to extract features of video contents. We set the learning rate of the policy network as 10^{-5} 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.

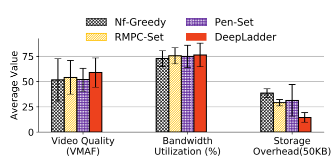
	Simulator (ms)	*In situ* (ms)			Acc.
	Execution Time	Trans Time	Measuring Quality	Total Time	
CBR	**0.048**	2010	5234	7243	98.83%
VBR	**0.0028**	2186	5186	7368	100%



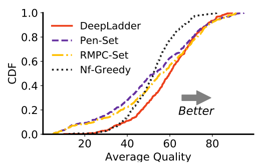
(a) HSDPA, by videos



(b) HSDPA, by network traces.



(c) FCC, by videos



(d) FCC, by network traces.

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%.

Baselines. 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.

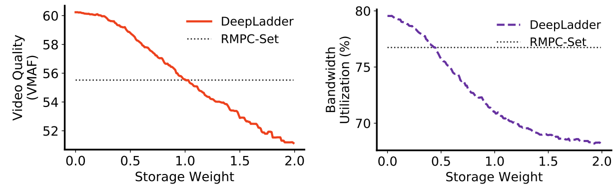


Figure 6: DeepLadder with different storage weight w.

▹ ***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

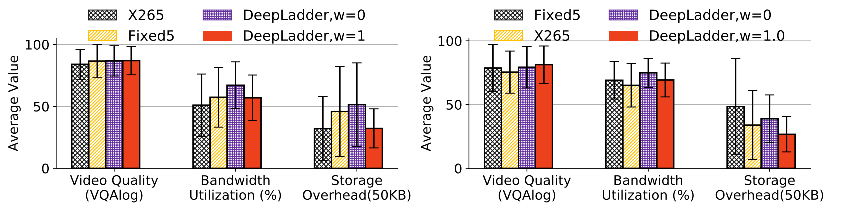


Figure 7: Comparing DeepLadder-VBR with existing bitrate optimization approaches on the Oboe and HDFS network traces.

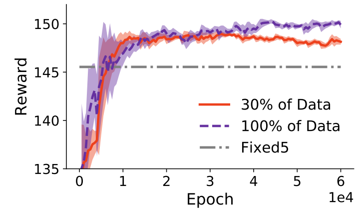


Figure 8: Training DeepLadder-VBR with 30% and 100% of videos.

and bandwidth utilization of DeepLadder with different w. Results are collected under the HSDPA dataset. Meanwhile, we plot the dashed lines to mark the performance of RMPC-Set. We find that the storage weight w can effectively control the balance among the given metrics: DeepLadder will preserve the average video quality instead of bandwidth utilization if the storage weight is rather high. Another interesting observation is that we set $w \in [0, 1)$ during the training, while DeepLadder even works well in $w \in [1, 2)$. Such observation demystifies that DeepLadder has already learned the correlation between the weight and the performance, and generalized a strategy that has a good extrapolation performance.

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.

$$VQA_t^{log} = 75.5 - \frac{65.4}{1 + e^{37.37 \times (SSIM_t - 0.93)}} + 24.4 SSIM_t \tag{3}$$

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.

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

Process(ms)	Video Extraction	Inference Timespan	Trans. Time	Extra Cost (%)
DeepLadder@CPU	439	3	12060	**3.67**
Netflix@CPU	-	-	39937	**332.0**
DeepLadder@GPU	94	12	3407	**3.11**
Netflix@GPU	-	-	11849	**347.8**

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 CPU2.2GHz 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 4× 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 (§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).

REFERENCES

[1] 2019. HTTP Live Streaming. https://developer.apple.com/streaming/. (2019).

[2] 2019. Youtube. (2019). https://www.youtube.com

[3] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

[4] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, et al. 2018. Oboe: auto-tuning video ABR algorithms to network conditions. In *SIGCOMM 2018*. ACM, 44–58.

[5] Pedro A Amado Assuncao and I Ghanbari. 1997. Optimal transcoding of compressed video. In *Proceedings of International Conference on Image Processing*, Vol. 1. IEEE, 739–742.

[6] Abdelhak Bentaleb, Ali C Begen, Saad Harous, and Roger Zimmermann. 2018. A distributed approach for bitrate selection in HTTP adaptive streaming. In *Proceedings of the 26th ACM international conference on Multimedia*. 573–581.

[7] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. 2018. A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP. *IEEE Communications Surveys & Tutorials* (2018).

[8] Chao Chen, Yao-Chung Lin, Steve Benting, and Anil Kokaram. 2018. Optimized transcoding for large scale adaptive streaming using playback statistics. In *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 3269–3273.

[9] Cisco. 2017. Cisco Visual Networking Index: Forecast and Methodology, 2016-2021. (2017). https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[10] Jan De Cock, Zhi Li, Megha Manohara, and Anne Aaron. 2016. Complexity-based consistent-quality encoding in the cloud. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 1484–1488.

[11] Jan De Cock, Aditya Mavlankar, Anush Moorthy, and Anne Aaron. 2016. A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications. In *Applications of Digital Image Processing XXXIX*, Vol. 9971. International Society for Optics and Photonics, 997116.

[12] FFmpeg. 2020. FFmpeg. https://ffmpeg.org. (2020).

[13] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella. 2017. D-DASH: A Deep Q-Learning Framework for DASH Video Streaming. *IEEE Transactions on Cognitive Communications and Networking* 3, 4 (Dec 2017), 703–718. https://doi.org/10.1109/TCCN.2017.2755007

[14] Guanyu Gao and Yonggang Wen. 2016. Morph: A fast and scalable cloud transcoding system. In *Proceedings of the 24th ACM international conference on Multimedia*. 1160–1163.

[15] Antonio Gulli and Sujit Pal. 2017. *Deep learning with Keras*. Packt Publishing Ltd.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[17] Alain Hore and Djemel Ziou. 2010. Image quality metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*. IEEE, 2366–2369.

[18] Angeliki V Katsenou, Joel Sole, and David R Bull. 2019. Content-gnostic Bitrate Ladder Prediction for Adaptive Video Streaming. In *2019 Picture Coding Symposium (PCS)*. IEEE, 1–5.

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Jani Lainema, Frank Bossen, Woo-Jin Han, Junghye Min, and Kemal Ugur. 2012. Intra coding of the HEVC standard. *IEEE transactions on circuits and systems for video technology* 22, 12 (2012), 1792–1801.

[21] Jean Le Feuvre, Jean-Marc Thiesse, Matthieu Parmentier, Mickaël Raulet, and Christophe Daguet. 2014. Ultra high definition HEVC DASH data set. In *Proceedings of the 5th ACM Multimedia Systems Conference*. 7–12.

[22] Ehud Lehrer and Rann Smorodinsky. 2000. Relative entropy in sequential decision problems. *Journal of Mathematical Economics* 33, 4 (2000), 425–439.

[23] Hongzi Mao. 2017. Pensieve-traces. (Jul 2017). https://www.dropbox.com/sh/ss0zs1lc4cklu3u/AAB-8WC3cHD4PTtYT0E4M19Ja?dl=0

[24] Hongzi Mao, Ravi Netravali, Mohammad Alizadeh, et al. 2017. Neural adaptive video streaming with pensieve. In *SIGCOMM 2017*. ACM, 197–210.

[25] NVIDIA. 2020. GPU-accelerated video processing integrated into the most popular open-source multimedia tools. (2020). https://developer.nvidia.com/ffmpeg

[26] Yanyuan Qin, Shuai Hao, Krishna R Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue. 2018. ABR streaming of VBR-encoded videos: characterization, challenges, and solutions. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 366–378.

[27] Jason J Quinlan and Cormac J Sreenan. 2018. Multi-profile ultra high definition (UHD) AVC and HEVC 4K DASH datasets. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 375–380.

[28] Reza Rassool. 2017. VMAF reproducibility: Validating a perceptual practical video quality metric. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on*. IEEE, 1–2.

[29] Fixed Broadband Report. 2016. Raw Data Measuring Broadband America 2016. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016. (2016). [Online; accessed 19-July-2016].

[30] Yuriy Reznik, Xiangbo Li, Karl Lillevold, Robert Peck, Thom Shutt, and Peter Howard. 2020. Optimizing Mass-Scale Multi-Screen Video Delivery. *SMPTE Motion Imaging Journal* 129, 3 (2020), 26–38.

[31] Yuriy A Reznik, Xiangbo Li, Karl O Lillevold, Abhijith Jagannath, and Justin Greer. 2019. Optimal Multi-Codec Adaptive Bitrate Streaming. In *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 348–353.

[32] Yuriy A Reznik, Karl O Lillevold, Abhijith Jagannath, Justin Greer, and Jon Corley. 2018. Optimal design of encoding profiles for abr streaming. In *Proceedings of the 23rd Packet Video Workshop*. 43–47.

[33] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 114–118.

[34] Werner Robitza. 2017. CRF Guide. (2017). https://slhck.info/video/2017/02/24/crf-guide.html

[35] Silvia Rossi, Cagri Ozcinar, Aljosa Smolic, and Laura Toni. 2020. Do Users Behave Similarly in VR? Investigation of the User Influence on the System Design. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 2 (2020), 1–26.

[36] SandDrive. 2020. COVID-19 Global Internet Phenomena Report. (2020). https://www.sandvine.com/phenomena

[37] Hamid R Sheikh, Muhammad F Sabir, and Alan C Bovik. 2006. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on image processing* 15, 11 (2006), 3440–3451.

[38] Kevin Spiteri, Ramesh Sitaraman, Daniel Sparacio, et al. 2018. From theory to practice: improving bitrate adaptation in the DASH reference player. In *MMSys 2018*. ACM, 123–137.

[39] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[40] Yuan Tang. 2016. TF. Learn: TensorFlow's high-level module for distributed machine learning. *arXiv preprint arXiv:1612.04251* (2016).

[41] Laura Toni, Ramon Aparicio-Pardo, Karine Pires, Gwendal Simon, Alberto Blanc, and Pascal Frossard. 2015. Optimal selection of adaptive streaming representations. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11, 2s (2015), 1–26.

[42] Usc-Nsl. 2018. USC-NSL/Oboe. (Oct 2018). https://github.com/USC-NSL/Oboe

[43] Vivek Veeriah, Junhyuk Oh, and Satinder Singh. 2018. Many-goals reinforcement learning. *arXiv preprint arXiv:1806.09605* (2018).

[44] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. 2019. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–37.

[45] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[46] Daniel Weinberger. 2015. Video Bitrates for Streaming. (2015). https://bitmovin.com/video-bitrate-streaming-hls-dash/

[47] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic forecasts achieve high throughput and low delay over cellular networks. (2013), 459–471.

[48] x265.org. 2015. The x265 website. https://x265.org/. (2015).

[49] Huaifei Xing, Zhichao Zhou, Jialiang Wang, Huifeng Shen, Dongliang He, and Fu Li. 2019. Predicting Rate Control Target Through A Learning Based Content Adaptive Model. In *2019 Picture Coding Symposium (PCS)*. IEEE, 1–5.

[50] Yanling Xu, Yueqiang Lin, and Chenfeng Yu. 2020. Rate-Distortion Cost Estimation Model Based on Cauchy Distributions for HEVC Encoder. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Vol. 1. IEEE, 436–440.

[51] Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. 2019. Mastering Complex Control in MOBA Games with Deep Reinforcement Learning. *arXiv preprint arXiv:1912.09729* (2019).

[52] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *SIGCOMM 2015*. ACM, 325–338.

[53] Anatoliy Zabrovskiy, Christian Feldmann, and Christian Timmerer. 2018. Multi-codec DASH dataset. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 438–443.

[54] Hui Zhang, Xiuhua Jiang, and Xiaohua Lei. 2015. A method for evaluating QoE of live streaming services. *international Journal of computer and electrical engineering* 7, 5 (2015), 296.