RL-AFEC: Adaptive Forward Error Correction for Real-time Video Communication Based on Reinforcement Learning

Ke Chen
New York University
New York, USA
kc4067@nyu.edu

Han Wang
New York University
New York, USA
hw2435@nyu.edu

Shuwen Fang
New York University
New York, USA
sf3405@nyu.edu

Xiaotian Li
New York University
New York, USA
xl3399@nyu.edu

Minghao Ye
New York University
New York, USA
my1706@nyu.edu

H. Jonathan Chao
New York University
New York, USA
chao@nyu.edu

ABSTRACT

Real-time video communication is profoundly changing people's lives, especially in today's pandemic situation. However, packet loss during video transmission degrades reconstructed video quality, thus impairing users' Quality of Experience (QoE). Forward Error Correction (FEC) techniques are commonly employed in today's audio and video conferencing applications, such as Skype and Zoom, to mitigate the impact of packet loss. FEC helps recover the lost packets during transmissions at the receiver side, but the additional bandwidth consumption is also a concern. Since network conditions are highly dynamic, it is not trivial for FEC to maintain video quality with a fixed bandwidth overhead. In this paper, we propose RL-AFEC, an adaptive FEC scheme based on Reinforcement Learning (RL) to improve reconstructed video quality with an aim to mitigate bandwidth consumption for different network conditions. RL-AFEC learns to select a proper redundancy rate for each video frame, and then adds redundant packets based on the frame-level Reed-Solomon (RS) code. We also implement a novel packet-level Video Quality Assessment (VQA) method based on Video Multimethod Assessment Fusion (VMAF), which leverages Supervised Learning (SL) to generate video quality scores in real time by only extracting information from the packet stream without the need of visual contents. Extensive evaluations demonstrate the superiority of our scheme over other baseline FEC methods.

CCS CONCEPTS

• **Information systems → Multimedia streaming**.

KEYWORDS

Real-time Video Communication, Forward Error Correction, Reinforcement Learning, Video Quality Assessment

MMSys '22, June 14–17, 2022, Athlone, Ireland
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9283-9/22/06…$15.00
https://doi.org/10.1145/3524273.3528184

ACM Reference Format:
Ke Chen, Han Wang, Shuwen Fang, Xiaotian Li, Minghao Ye, and H. Jonathan Chao. 2022. RL-AFEC: Adaptive Forward Error Correction for Real-time Video Communication Based on Reinforcement Learning. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3524273.3528184

1 INTRODUCTION

Internet video today dominates Internet traffic and is expected to account for more than 80% of all consumer Internet traffic by 2021 [15]. Starting from early 2020, the unexpected world-wide pandemic accelerated this trend since more and more people are relying on video conferencing to work/study remotely. According to research by Stanford University, work-from-home employees now account for more than two-thirds of U.S. economic activity [4]. For example, Facebook's employees can continue working remotely after the pandemic [3]. Twitter also announced the decision to make "working from home" a permanent choice [17]. In this situation, real-time conferencing applications allowed us to stay in touch with others and became essential in people's daily lives.

However, packet loss during video transmission can be a potential threat to video quality. From a recent study, when 1% or more packets are lost during transmission, video quality degradation becomes noticeable by video conferencing participants, and the video quality is generally unacceptable at a 5% packet loss rate [1]. In today's network, packet loss usually happens due to network issues like delay/jitter and network congestion [16]. For instance, if the packets arrive late at the jitter buffer of the receiver, or the links carry more data than they can handle, packets are discarded and considered as packet loss. Since the lost packets may contain useful information (e.g., motion vectors) for a video decoder at the receiver to reconstruct original videos, it would negatively affect decoder performance by generating video artifacts and degrading video quality [7].

To address the above mentioned issue, video-on-demand services such as YouTube and Netflix adopt Transmission Control Protocol (TCP) to retransmit lost packets [30] by taking advantage of the fact that video-on-demand services are delay insensitive and lost packets can be retransmitted. However, video live-streaming or video conferencing call applications carried over Real-time Transport Protocol (RTP) and User Datagram Protocol (UDP) [36] cannot afford the delay from retransmitting lost packets. Instead, they normally recover lost packets in real time at the receiver using Forward Error Correction (FEC) techniques.

FEC is commonly used in today's audio and video conferencing applications, such as Skype [13] and Zoom [21], to detect and correct a limited number of errors caused by packet loss. Basically, FEC adds additional redundant packets that are generated from original source packets by the FEC encoding process. When source packets are lost during transmission, the receiver can still reconstruct them using the redundant packets through the FEC decoding process. Several heuristic FEC algorithms take historical information as an estimate of future network condition to determine the redundancy rates [5], [27], [10]. However, since network conditions are fluctuating and packet loss rates change with time, such estimate could be inaccurate in dynamic network environments, which may result in unrecovered packet losses or wasted bandwidth. It is challenging to devise an effective FEC scheme adaptive to fluctuating packet loss rates while saving as much bandwidth as possible.

Reinforcement Learning (RL) with integration of Deep Learning (DL) techniques has demonstrated great potential in applications such as playing games [26], [45], Computer Go [33], adaptive video streaming [22], and traffic engineering [47], [46]. RL techniques are capable of learning adaptive policies through interactions with environments. In a specific task, the trained RL agent will properly react to the environment dynamics, with a purpose to maximize the accumulated reward defined in advance. This adaptive characteristic of RL can help to balance the trade-off between the requirements to recover the fluctuating packet loss and not to waste bandwidth for over-protection.

In this paper, we first propose a no-reference packet-level video quality assessment (PL-VQA) model by supervised learning (SL). After training, the PL-VQA model learns the mapping from some packet-level features to the corresponding Video Multimethod Assessment Fusion (VMAF) scores, a famous quantitative video quality metrics highly correlated with subjective results. The model can estimate the VMAF scores precisely without reference to the visual contents, neither original video nor received video, which is much faster than most existing video quality metrics and supports real-time operations. The estimation also serves as QoE metrics perceived by the client, a key component of the environment for the RL agent to learn a policy.

With the help of the PL-VQA model, we propose RL-AFEC, an RL-based approach that learns to select a proper redundancy rate automatically for each video frame to improve reconstructed video quality with low bandwidth consumption in dynamic network conditions. Here, the redundancy rate refers to the percentage of redundant packets required to recover the lost packets. RL-AFEC selects the redundancy rate at the receiver side and sends it back to the sender. The sender then uses the frame-level Reed-Solomon (RS) code [31] to generate redundant packets, which are then added to the source packet stream for transmission. Since it is not trivial to explore enough to learn a good policy in a very large action space, we design a discrete action pool with 10 different redundancy rates (10%, 20%, ..., 100%) to reduce the action space and also accelerate the convergence. We introduce the idea of *critical frames* with consideration that packet loss on different video frames do not have equivalent impact. Thus, the redundancy rates for critical frames are selected individually while non-critical frames share a same redundancy rate. To train RL-AFEC, we simulate a large video stream using RTP over the improved Gilbert-Elliot (GE) channel [11], [9]. To cope with video live-streaming/video conferencing call applications, RL-AFEC uses PL-VQA to estimate received video quality and determines the redundancy rates accordingly. After getting enough training in the simulation environment, the RL-AFEC model can select the appropriate redundancy rates, maximizing the long-term accumulated reward by balancing between the protection against packet loss and the waste of bandwidth.

The contributions of this paper are summarized as follows:

- We design an SL-based packet-level VQA method to evaluate received video quality in real time without the need of reconstructing visual contents.
- We propose an RL-based Adaptive FEC scheme to determine redundancy rates according to fluctuating network conditions with an aim to mitigate bandwidth consumption.
- We evaluate and compare RL-AFEC with the baseline methods by conducting extensive experiments on real-world video datasets. RL-AFEC can achieve good video quality for more than 95% of 1-second videos with as low as 40% additional bandwidth consumption, while the baseline models can only achieve 90% "good" quality videos with more than twice the bandwidth consumption of RL-AFEC.

The rest of the paper is organized as follows. Some background information and related works are introduced in Section 2 In Section 3 and Section 4, we discuss the design of our PL-VQA method and the RL-AFEC scheme, respectively. Section 5 evaluates the performance of RL-AFEC and presents simulation results. Section 6 concludes the paper.

2 BACKGROUND AND RELATED WORKS

In this section, we introduce background information and some related works.

2.1 Video structures

Today's videos are encoded and decoded using a Group of Pictures (GoP) frame structure that consists of three frame types known as I-frames, P-frames, and B-frames. An I-frame is an independently encoded reference frame. Each GoP only contains one I-frame and always starts with an I-frame. P-frames use information from the previous frames and contain only the motion changes relative to the previously encoded frames. B-frames use both the previous and following frames as references for encoding. The higher ratio of P-frames and B-frames used instead of I-frames in a video, the smaller the encoded video size, and the less required bandwidth when streaming. Considering packet loss, the impact of packet loss in an I-frame or a P-frame will propagate within the GoP, and cause potential distortion for the remaining frames. When there are packet losses in a B-frame, such impact does not propagate since B-frames do not provide a reference to any other frames. A new I-frame of next GoP will remove all the accumulated error in the current GoP. However, both the encoding and decoding of the B-frames requires the future frames as reference, leading to extra delays and is thus not favorable in real-time applications.

In this paper, all videos are encoded using H.264 by setting Frames per Second (FPS) and the GoP size to 30 while disabling B-frames to minimize the overall delay. The fixed FPS and GoP is to simulate a general real-time video communication scenario, where the motion between frames is neither too dynamic nor too stable.

2.2 Video Quality Assessment (VQA)

Objective video quality assessment, which uses statistical models to approximate subjective user-perceived video quality ratings provided by users (e.g. Mean Opinion Scores, MOS), can be divided into two main categories: Full-Reference (FR) and No-Reference (NR), based on the availability of the reference video.

FR video metrics evaluate video quality by performing frame-by-frame comparison between the to-be-tested videos and the original videos. Among all the FR video metrics, Video Multimethod Assessment Fusion (VMAF) developed by Netflix can predict the video quality score highly correlated with subjective user-perceived ratings, and has been widely adopted in the community and industry [19]. VMAF extracts spatial and temporal information from the to-be-tested and the original videos, and predicts a video quality based on the combination of multiple basic quality metrics. The idea is that each basic quality metric may have its own strengths and weaknesses with respect to the source content characteristics, type of artifacts, and degree of distortion. By integrating the basic metrics into a more powerful metric with a Support Vector Machine (SVM) regressor, the fusion metric could preserve all the advantages of the individual metrics, and hence deliver a more accurate video quality score.

Although FR metrics like VMAF can measure the video quality precisely, the additional computational cost and the requirement of the presence of the original videos make FR metrics impractical for real-time video communications. This is where NR metrics stands out. Because of no requirement on the original video as reference, NR metrics are more feasible but more challenging to obtain when assessing video quality at the receiver side.

Based on the information required, most existing NR video quality assessments can be categorized into two types: pixel-based methods and bitstream-based methods. Pixel-based methods evaluate the visual content of the decoded video based on pixel information and video structures. For example, Naturalness Image Quality Evaluator (NIQE) [24] is based on spatial-specific natural scene statistic features that are derived from local image patches. A more advanced algorithm, Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) [23], constructs point-wise features from local normalized luminance signals with the help of a natural image model to estimate the image naturalness. Both schemes use an image quality evaluator and compute the average value along the frames to obtain a video quality score. Alternatively, bitstream-based video quality assessments use features derived from the information of the video packet streams, including packet headers, motion vectors, and quantization parameters [12]. Bitstream-based methods require neither the original videos nor the decoded videos, thus yielding a higher computation efficiency compared to pixel-based methods. For the video applications with the delay as a crucial factor affecting the QoE of the service (i.e., real-time video conferencing), the extra delay caused by the computation overhead of a pixel-based NR video quality metric could undermine the performance. In contrast, a bitstream-based NR video quality assessment approach could help reduce the computation delay.

2.3 Forward Error Correction (FEC)

Several proposed FEC schemes are based on a single or multiple consecutive GoPs. For instance, Baccaglini et al. [2] employs GoP-level Reed-Solomon (RS) code, where all frames of an entire GoP are used to generate one RS block. Xiao et al. [41] proposed sub-GoP level FEC coding, where the length of sub-GoP is dynamically changed according to network conditions and encoding parameters. Authors in [42] generate RS parity packets using video packets of the current frame and the previous frames of the current GoP. The lost packets can be recovered by jointly solving the combinations of the parity-check equations, which requires additional computational overhead. However, these prior works are not applicable in real time since the receiver must wait for an entire GoP or sub-GoP to recover a frame, resulting in unacceptable delays.

Some FEC schemes adopt frame-level strategies. For instance, Yang et al. [43] and Wu et al. [38] explored frame-level FEC by maximizing the expected number of received frames under a total FEC bitrate constraint, while Kurdoglu et al. [18] implemented FEC by maximizing a proposed perceptual video quality model through jointly adapting the frame rate and FEC. Several papers [35], [40], [39] employed FEC TCP-based video streaming to mitigate frequent video playout pausing caused by TCP congestion control, which regulates transmission rate and thus could exhaust the receiver's play-out buffer. Almost all the prior work allocated redundant packets by optimizing a predefined QoE objective function based on network statistics such as packet loss rate and delay.

Recently, several schemes using ML to optimize FEC strategies have emerged. DeepRS [6] can predict the number of packets that might be lost in the next block by training an LSTM model with historical loss patterns. However, the model did not take video characteristics (i.e., frame size and motion estimation) into consideration when making decisions. On the other hand, these video characteristics could be important for the ML model to obtain a good result, since different types of videos and different video frames are not of equal importance regarding their impact to the video quality. The authors in [14] proposed a decision tree-based model that considers network conditions and video content information when applying FEC. However, it requires the actual video content as part of the input, which is not feasible in real-time video communications. Moreover, the model does not consider bandwidth consumption and the decisions are made based on the coarse-grained six levels of video quality (i.e., Poor, Bad, Fair, Good, Very good, and Excellent). To the best of our knowledge, no prior work has studied RL-based FEC for real-time video communications to improve video quality and minimize the additional bandwidth used for FEC at the same time.

2.4 Software-Defined Wide Area Network (SD-WAN)

The emerging SD-WAN technology has benefited enterprises by reducing costs and improving network performance [44], [37], [29].

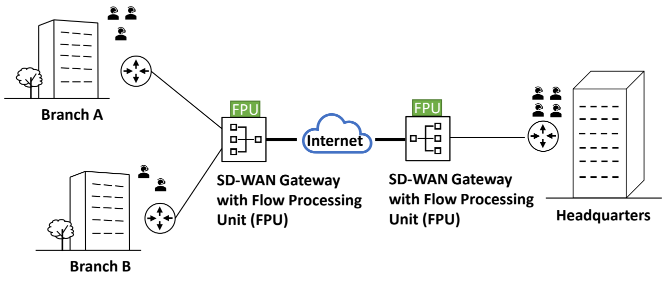

Branch A
FPU
Internet
FPU
SD-WAN Gateway with Flow Processing Unit (FPU)
SD-WAN Gateway with Flow Processing Unit (FPU)
Headquarters
Branch B

Figure 1: Illustration of an SD-WAN architecture.

SD-WAN solutions offer enterprises the flexibility to access multi-cloud services and to offer new applications and services. By taking advantages of the fact that various virtual network functions for network security, traffic engineering, among others, have been implemented at the SD-WAN gateways, we propose to implement an effective and efficient Adaptive FEC (AFEC) scheme for real-time video streams at the gateways. Our proposed AFEC scheme could determine the redundancy rate for each video clip periodically to maximize the accumulated reward estimated by VQA for each flow at the destination gateway. To meet the low-delay requirement in real-time scenario, we design a Flow Processing Unit (FPU) as a built-in module at the SD-WAN gateway, as shown in Figure 1. When video flows reach an egress SD-WAN gateway, FPU will identify each flow and collect the video features (i.e. packet loss rate per frame, frame size, and motion estimation) from the RTP stream. The features are used to determine the redundancy rate based on RL-AFEC algorithm (to be described in Section 4). The redundancy rate for a certain video flow is then sent back to the corresponding ingress SD-WAN gateway, which will perform FEC encoding accordingly for the video flow.

3 PACKET-LEVEL VIDEO QUALITY ASSESSMENT

In this section, we describe the design of our proposed Packet-Level Video Quality Assessment (PL-VQA) method, which is one of the key components in RL-AFEC to perform real-time video quality assessment.

3.1 PL-VQA Design

In this section, we proposed a packet-level no-reference video quality assessment method named PL-VQA based on Supervised Learning (SL). We extracted representative features from a video packet stream as the input every second. A four-layer deep neural network is used to learn a mapping between the input packet-level feature to the ground truth video quality score produced by VMAF algorithm.

The pipeline for generating the training and testing dataset is shown in Figure 2. For the first step, we took original videos from our video collection as described in Section 5 and added different levels of impairment (packet loss) to them to get the degraded videos. The average packet loss rates are controlled by the Gilbert-Elliot (GE) channel, which is a widely used, two-state Markov model for simulating random losses. Each degraded video with reference to

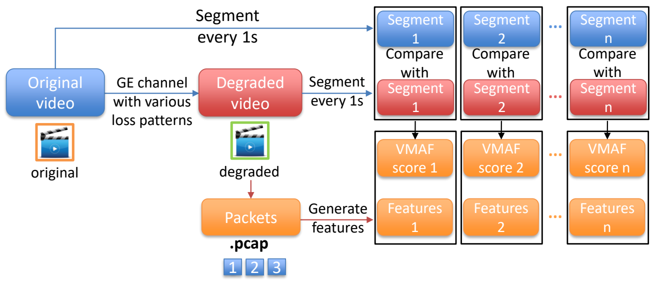


Figure 2: Overall approach for generating training and testing data.

the original video represents a received video with the corresponding packet loss rate.

To capture packet-level information, we set up two Virtual Machines (VMs), one as the sender host and one as the receiver host. Videos are encoded and sent through two VMs to retrieve the video packet streams at the receiver side. The protocol that we use to transmit the video is the RTP, which is broadly used in communication systems including streaming media, IP telephony, and video conferencing. Each RTP packet header contains information including frame types, sequence number, and segmentation identifier. Those statistics are essential for constructing the packet-level features.

Once we obtained the degraded video with reference to the original video and the video packet capture, we are able to construct our training and testing data.

3.1.1 *Packet-level Feature Extraction.*

Video packet headers include various descriptive data, such as sequence number, video frame type, payload size, and segmentation identifier. From the feature design perspective, we must reserve the representative and crucial properties of each video. Good features should not only reflect various levels of the degradation but also characterize different video types. In general, our packet-level feature consists of three parts: packet loss rate per frame, frame size in bytes, and motion estimation factor.

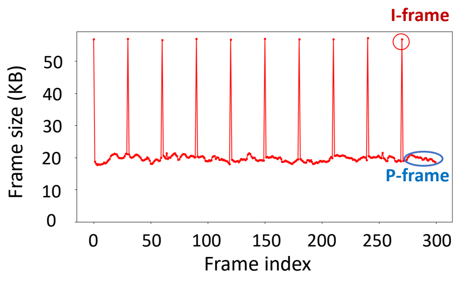
Given a 1-second video segment corresponding to our smallest data point, there are a total of 30 frames. Based on the sequence number and segmentation identifier, we are able to rule out which packets forming a frame are lost during transmission. Consequently, we computed the number of bytes in the lost packets and obtained the packet loss rate[1] within each frame using Equation 1.

$$packet\ loss\ rate = \frac{\#\ of\ bytes\ lost\ per\ frame}{\#\ of\ total\ bytes\ per\ frame} \tag{1}$$

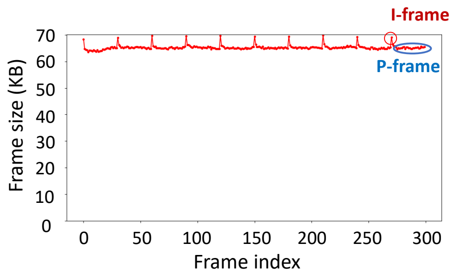
For each frame, we will get a real number in the range from 0 to 1. For the 1-second video, we will get a feature vector with a size of 30 as the first part since the videos are encoded in 30 frames per second.

For the second part, the number of bytes within each frame can be easily obtained from the RTP header as well. The denotation is

[1]We use byte loss rate to approximate packet loss rate since most packets are of the same size.



(a) Akiyo



(b) Hall

Figure 3: Comparison of frame size between two video sequences: Akiyo and Hall.

shown in Equation 2. We then normalized the frame size in bytes to the range from 0 to 1, and the size of the feature vectors is also 30.

$$frame\ size = \#\ of\ total\ bytes\ per\ frame \tag{2}$$

For the last part, we proposed a motion estimation factor to measure how much motion is within each GoP, as shown in Equation 3. We came up with this equation from the observation of the size of frames in different video sequences. For example, Figure 3 shows the size of each frame in two videos named Akiyo and Hall. Akiyo is a slow-motion video with one newscaster reporting news in front of a static background. In contrast, Hall is a relatively high-motion video which comes from a hall monitor with random people walking in and out of office. From the figure, we can see that the slow-motion videos usually contain a smaller size of P frames compared with high-motion videos. Therefore, we use the ratio of P-frame size to I-frame size to estimate the motion.

$$motion\ estimation\ factor = \frac{P\text{-}frame\ size}{I\text{-}frame\ size} \tag{3}$$

To conclude the packet-level feature extraction, we concatenated all three parts of the features: packet loss rate per frame, frame size, and motion estimation factor. In total, we will get a feature vector of size 90 as input to the PL-VQA for one GoP that is corresponding to a 1-second video segment. The values in each dimension of the feature vector are in the range (0,1]. The packet-level features can capture different levels of distortion and indicate the characteristics of different video types.

3.1.2 VMAF Score Collection. Both the degraded and the original video are divided into video segments with a length of 1 second. We then took the pairs of segments as the input to the VMAF model to get the VMAF score. In this step, we access VMAF using the FFmpeg tool [34]. Note that the finest granularity of VMAF is the per-frame score; therefore, the VMAF scores for the 1-second video segments are derived by taking the per-frame scores and applying a temporal pooling to get the per-second scores.

3.1.3 Video Quality Score Prediction. We adopted SL regression method to train a mapping from the packet-level feature to the ground truth VMAF. For the regressor model, we used a four-layer fully-connected deep neural network. The input layer size is set to (90,) to match our packet-level feature vector size. Subsequently, we attached two fully-connected hidden layers with the number of hidden nodes in both layers set to 180. From the output, we obtain the predicted video quality score.

3.2 Evaluation

The evaluations of our PL-VQA model include two parts. First, we evaluate the performance of our model in terms of prediction accuracy with respect to the ground truth VMAF scores. Second, we evaluate the computation time of our video quality assessment method compared to other full-reference and no-reference video quality assessment method.

3.2.1 Prediction Accuracy. To take an overall look at the prediction accuracy performance, we adopted two evaluation metrics called Mean Absolute Error (MSE) and Mean Percentage Error (MPE), as shown in the following equations:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4}$$

$$MPE = \frac{100\%}{N}\sum_{i=1}^{N}\frac{y_i - \hat{y}_i}{y_i} \tag{5}$$

where y_i and $\hat{y}_i$ refer to predicted VMAF scores and ground truth VMAF scores separately. Table 1 shows the performance in both the training dataset and testing dataset. From this table, we can see that the prediction accuracy in terms of MSE and MPE is satisfactory in both datasets.

Table 1: Prediction Accuracy Evaluation

	MSE	MPE(%)
Training Dataset	5.27	7.42%
Testing Dataset	6.04	8.17%

To get a better sense about our PL-VQA model's performance and to get a more straightforward look, we use the plot in Figure 4 to demonstrate the correlation between the predicted scores and the ground truth scores. The x-axis represents the ground truth VMAF quality scores, while the y-axis represents the predicted video quality scores from PL-VQA. The red line across the plot denotes the optimal situation where the predicted scores equal to the ground truth scores. Meanwhile, each point corresponds to a per-video score located by the ground truth and the predicted value. Different colors of points represent different original videos as the reference. Different points sharing the same color represent the different levels of distortion. From this plot, we can see that

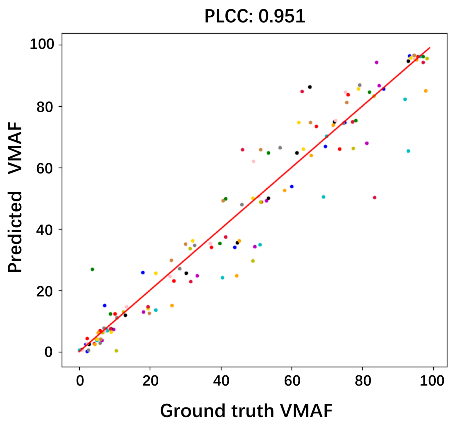


Figure 4: Correlation between Predicted Score and Ground Truth Score.

our data points are closely distributed around the optimal line. We also compute the Pearson linear correlation coefficient (PLCC), a measurement of linear correlation between two sets of data ranges from -1 to 1. PLCC equals 1 would mean a perfect linear correlation. The closer it is to 1, the better the linear correlation they share. In our case, the PLCC is 0.951, which implies an excellent correlation for the PL-VQA model.

Table 2: Computational Time for a 1-second Video Clip

Model	Inference Time
PL-VQA (Ours)	5ms
BRISQUE (Pixel-based NR VQA)	>1s
VMAF (FR Video Quality Metric)	>20s

3.2.2 Computation Cost. For the second part of the evaluation, we focus on the actual computation time for our proposed PL-VQA and other video quality assessment methods. Computation complexity is crucial for monitoring QoS and QoE for many real-time video applications. Generally speaking, the additional computational delay caused by most existing video quality assessment methods would make them unsuitable for the task. Table 2 shows the computational time to obtain a video quality score for a 1-second video segment using different video quality assessment methods. We can see that our proposed method demonstrates huge advantages in terms of computation cost, making it the only feasible video quality assessment method for real-time video applications. This is because we implemented our PL-VQA model with reduced input size and complexity, making it much more computationally efficient.

4 RL-AFEC DESIGN

In this section, we provide the details of designing the RL-AFEC model. We discuss the design of three components in reinforcement learning: state, action, and reward, followed by the description of the model architecture and the loss function.

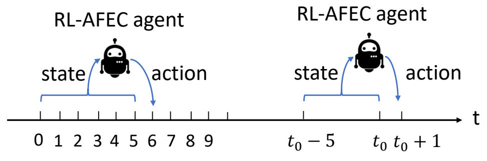


Figure 5: The general idea of state design. Each state contains information of video frames from the past 5 seconds.

4.1 State

In every second, the RL agent takes a state as input, which contains video information of the past adjacent 5 seconds, and outputs an action (i.e., FEC redundancy rate) to be applied to the video transmission in the next second. As shown in Figure 5, the action selected for performing FEC encoding from t_0 to $t_0 + 1$ is based on information from $t_0 - 5$ to t_0. In every second, three types of information are extracted: *packet loss rate per frame*, *frame size*, and *motion estimation*, which are the same as the input of the PL-VQA model. Similar to PL-VQA, every second's feature is a 90-dimensional vector. Every five consecutive feature vectors are combined into a 90×5 matrix and used as an input state.

4.2 Action

The action of our model is to select a redundancy rate to be used for each frame to add corresponding redundant packets. The main challenge for this is the huge action space, which may cause the convergence of the model to be impaired [8].

To address this issue, first we predefine an action pool that contains 10 different redundancy rates (10%, 20%, ..., 100%), instead of using the continuous action space. In every second, the agent receives an input state, and selects one of the 10 redundancy rates for each of the 30 frames. For example, if a frame is carried by 10 source packets and the agent chooses a 20% redundancy rate for this frame, then 2 more redundant packets will be added. We employ per-frame Reed-Solomon (RS) code to perform FEC where all packets belonging to the same frame serve as a block.

Even with the much smaller discrete action space, the agent still needs to select a redundancy rate for each of the 30 frames out of the 10 different redundancy rates every second, leading to an action space as large as 10^{30}. To keep on reducing the action space, we take video structure into account. As explained in Section 2, any error in a frame will be propagated to the end of the GoP. The earlier the corrupted frame is, the greater the impact would be. As a result, the model should pay more attention to a certain number (i.e. K) of frames in the front of a GoP. We call them *critical frames*.

Through extensive action refinement, the RL-AFEC model now chooses one of the 10 redundancy rates (10%, 20%, . . . , 100%) for each of the K critical frames individually and selects a redundancy rate for all the remaining non-critical frames. Figure 6 provides an illustrative example to explain the idea. Assuming that the first 4 frames are critical frames and the remaining 26 frames are non-critical frames, the agent selects different redundancy rates (50%, 30%, 40%, and 60%) for each of the critical frames but uses a fixed redundancy rate (20%) for all non-critical frames. Given K critical

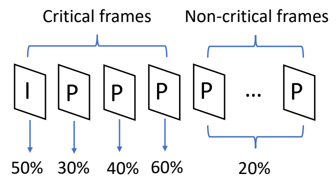


Figure 6: An example of action design with critical frames.

frames, the action space is now reduced to 10^{K+1}. We will discuss how to determine the value of K in Section 5.

4.3 Reward

Our goal is to maintain high video quality with minimum bandwidth consumption. Therefore, there are two factors considered in the reward function. One is video quality estimated in VMAF score based on our PL-VQA model, as elaborated in Section 3; the other is bandwidth wasted, which is defined as follows.

$$BW_{wasted}(i) = \begin{cases} pkt_{add}(i) - pkt_{lost}(i), & \text{if loss recovered} \\ pkt_{add}(i), & \text{otherwise} \end{cases} \tag{6}$$

Here, $pkt_{add}(i)$ refers to the number of redundant packets added to the i-th frame, whereas $pkt_{lost}(i)$ refers to the number of lost packets in the i-th frame. We designed the reward function based on the property of the RS code. For $RS(n,k)$, if $n-k$ redundant packets are added to k source packets, then as long as the receiver receives k packets (regardless source packets and redundant packets), the original k source packets can be recovered. In other words, if the lost packets can be recovered, the number of redundant packets should not be lower than the number of lost packets. Therefore, the redundant packets that are not used for RS decoding can be considered as wasted packets. If the lost packets cannot be recovered due to lack of redundant packets, then all the added redundant packets are useless and the bandwidth is wasted. For example, if one frame is carried by five packets, we use $RS(7,5)$ to perform FEC encoding so that two more redundant packets are added. During transmission, if one of the source packets is lost, it can be recovered since we have provided enough redundancy packets. In this case, the bandwidth wasted is $2-1=1$ packet. However, if 3 packets are lost during transmission, it is not sufficient to recover the lost ones with two redundant packets. As a result, the bandwidth wasted would be two packets. When the number of added packets is exactly the same as the number of lost packets in the future, the bandwidth wasted would be zero.

The reason we consider bandwidth wasted instead of the total amount of bandwidth consumed is that we want the model to recover the lost packets with the minimum additional bandwidth. The lost packets can always be recovered if we add 100% redundancy rate all the time. However, if the loss rate is relatively low, a high redundancy rate would lead to unnecessary bandwidth waste. Therefore, it is important to determine how to limit additional bandwidth at a proper amount such that the redundant packets can be used to recover the lost packets without bandwidth wasted.

The general form of the reward function is:

$$r = VMAF - \alpha \cdot \sum_{i=1}^{30} BW_{wasted}(i), \tag{7}$$

where α is a scaling factor controlling the weight of the penalty term. Basically, models with a small alpha tend to consume more bandwidth compared to models with a large alpha.

The general form of the reward function above implies the relationship between video quality and consumed bandwidth. However, it cannot provide performance guarantee to achieve good video quality. In our design, we first ensure that the video quality score is above a preset target score regardless of bandwidth consumption. Once the target score can be achieved, we then consider constraining the bandwidth consumption. Thus, we refine the reward function into a piece-wise function as shown below.

$$r = \begin{cases} VMAF - VMAF_{target}, & \text{if } VMAF < VMAF_{target} \\ VMAF_{target} - \alpha \cdot \sum_{i=1}^{30} BW_{wasted}(i), & \text{otherwise} \end{cases} \tag{8}$$

When the $VMAF$ score is smaller than the $VMAF_{target}$, the reward is negative and solely relies on the $VMAF$ score. The closer to the target, the higher the reward. Once the VMAF score reaches the target, the reward becomes positive with an increasing penalty on bandwidth wasted, which is no longer dependent on VMAF score. We set the target score $VMAF_{target}$ equal to 80, which indicates 'good' video quality according to Netflix [20].

4.4 Model's Architecture

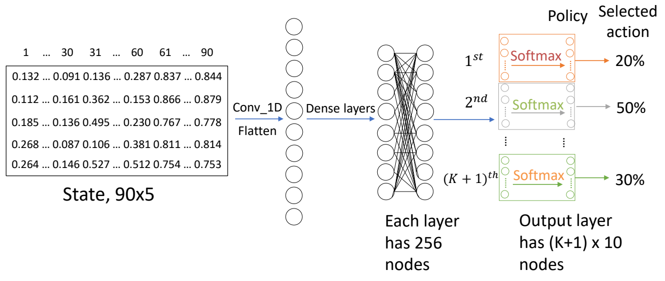


Figure 7: The architecture of RL-AFEC model.

We employ a pure policy approach for training RL-AFEC model. Figure 7 shows the architecture of our model. The state information, which is a 90×5 matrix, would be fed into a 1-D convolutional layer with 8 filters. After flattening, the intermediate result passes through 2 dense layers, where each layer has 256 nodes. At the end, it reaches the output layer. To select a specific redundancy rate for each critical frame, we implemented the softmax function independently on output nodes corresponding to each critical frame. As shown in Figure 7, different colored boxes represent different critical frames except the last one, which corresponds to the remaining non-critical frames. Each box contains 10 nodes that refer to 10 different redundancy rates in the action pool. After the softmax function, we obtain the policy, which is the probability that the model relies on to make decisions. In every second, the agent determines $K + 1$ actions, including K actions for the critical frames and 1 action for the remaining non-critical frames. The reference time of RL-AFEC is 2.14 ms on average and the selected redundancy accounts for the next whole seconds, which accounts for low computation overhead and it suitable for real-time operations.

4.5 Loss Function

In this subsection, we describe the loss function used to train the model. By minimizing the loss function, the goal is to find an optimal policy π (i.e., the probability of choosing each action). In Figure 7, there are $K + 1$ different colored boxes corresponding to different frames, and $K + 1$ actions are determined out of the boxes accordingly. Therefore, there are two types of loss functions: one is the loss function for each box (i.e., the loss function for selecting a single action out of 10 candidates), and the other is the loss function for the whole system (i.e., the loss function for selecting multiple actions, in our case $K + 1$ actions each time). We first illustrate the loss function design for selecting single action, and then extend it to multiple actions.

Table 3: Notations

N	the size of action pool
K	the number of critical frames
s_t	state at time t
a_t	action at time t
r_t	reward at time t
$\pi(a_t\|s_t)$	probability over all N actions in state s_t
$b(s_t)$	baseline for reducing gradient variance
$H(\pi(\cdot\|s_t;\theta))$	entropy of the policy

4.5.1 Loss Function for Selecting Single Action.

The agent takes a state s_t as input and outputs a probability distribution $\pi(a_t|s_t)$ over all possible actions in the action pool. The loss function is defined as follows:

$$loss = -\sum_t (\log \pi(a_{t_i}|s_t;\theta)(r_t - b(s_t))) \tag{9}$$

A baseline $b(s_t)$ is added to reduce gradient variance and thus speed up the convergence. In our method, we use the average reward for each state s_t as the baseline. $(r_t - b(s_t))$ indicates how much better a specific action is compared to the "average" for a given state s_t according to the policy. Intuitively, if $(r_t - b(s_t))$ is positive, $\pi(a_{t_i}|s_t;\theta)$ (i.e., the probability of choosing action a_{t_i}) is increased by minimizing the loss function. Otherwise, the probability is decreased. The net effect of Equation 9 is to reinforce actions that lead to higher rewards.

However, the above loss function cannot always lead to good convergence. If the model hits a good action that results in a positive reward at the first time, the model might reinforce this action too much (i.e., increase its probability to near 1). As a result, other actions will not get a chance to be chosen. It is possible that there exists another action that can result in a larger reward. This problem is called the lack of exploration. To ensure that the RL agent explores the action space adequately during training to discover good policies, the entropy of policy π is added to Equation 9. This technique improves the exploration by discouraging premature convergence to suboptimal deterministic policies [25]. Then, Equation 9 is modified as follows:

$$loss = -\sum_t (\log \pi(a_{t_i}|s_t;\theta)(r_t - b(s_t)) - H(\pi(\cdot|s_t;\theta))) \tag{10}$$

where H is the entropy of the policy.

4.5.2 Loss Function for Selecting Multiple Actions.

Equation 10 is used to evaluate the quality of action selected for one frame (i.e. a single box in the output layer in Figure 7). Recall that we have $K + 1$ actions to be determined at each time, where K is the number of critical frames. To design a loss function that characterizes the whole system, we simply sum up the loss functions over the number of actions. The final form of loss function is given below:

$$\begin{aligned} loss = \sum_{k=1}^{K+1} (-\sum_t (\log \pi(a_{t_i}^{(k)}|s_t;\theta)(r_{t_i} - b(s_t)) \\ - H(\pi(a_t^{(k)}|s_t;\theta)))), \end{aligned} \tag{11}$$

where $a_{t_i}^{(k)}$ refers to the action selected for the k-th critical frame ($k \le K$) at time t, while $a_{t_i}^{(K+1)}$ is the action selected for non-critical frames.

Then, the policy parameter θ is updated according to the following equation:

$$\begin{aligned} \theta \leftarrow \theta + \alpha \sum_{k=1}^{K+1} (\sum_t (\nabla_\theta \log \pi(a_t^{(k)}|s_t;\theta)(r_t - b(s_t)) \\ + \beta \nabla_\theta H(\pi({a_t}^{(k)}|s_t;\theta)))), \end{aligned} \tag{12}$$

where α is the learning rate for the policy network, while β controls the strength of the entropy regularization term.

5 EVALUATION

In the previous section, we elaborated on the design of the RL-AFEC scheme. This section presents the evaluation results of our method.

5.1 Dataset Generation

As shown in Figure 8, we collected 12 commonly used video sequences in YUV 4:2:0 format to generate our video dataset [32]. The videos cover different motion levels, including slow motion (e.g., akiyo and news), medium motion (e.g., foreman and container), and high motion (e.g., coastguard and stefan).

To create the dataset for RL training, we concatenated videos in a random order to simulate a large video stream on the Internet. The videos are transmitted in a simulation environment using RTP through the GE channel. The GE channel is a two-state Markov model, with transition probabilities from good-to-bad as α and that from bad-to-good as β, $\alpha, \beta \in (0, 1)$. The packet loss rate in the good-state is $\epsilon \in [0, 1)$, and in the bad-state is 1. The overall packet loss rate of a $GE(\alpha, \beta, \epsilon)$ is

$$PLR(\alpha, \beta, \epsilon) = \frac{\alpha}{\alpha + \beta} + \frac{\beta}{\alpha + \beta}\epsilon \tag{13}$$

To introduce various loss patterns to the video transmission process, we use a Hidden Markov Model (HMM) [28] to further

Figure 8: Video pool for generating training and test dataset. All videos are in CIF format and encoded using H.264 by setting Frames Per Second (FPS) and Group of Picture (GoP) size equal to 30.

represent the nature of the Internet packet loss pattern. Instead of solely using the classic GE channel, we concatenate the loss traces generated using the GE channel with 16 different sets of parameters (α, β, ϵ) that correspond to 16 different average loss rates (i.e., 0.1%, 1%, 2%, ..., 15%). We proposed an HMM with 16 states to control the generation of loss traces, where each state corresponds to an average loss rate used in the GE channel. Both the transition probability and emission probability obey the binomial distribution. We set the transition time interval to 5 seconds, which means the HMM transits to another set of GE parameters every 5 seconds. By using the HMM, we simulate the loss traces with similar patterns as real Internet loss traces in a long time span, and introduce a temporal correlation between the past and future loss patterns. Then, we can extract frame level information from the packet stream, such as packet loss rate per frame, frame size, and motion estimation as discussed in Section 3. Each entry in the dataset is a 90-dimensional vector that corresponds to a 1-second video clip or 30 frames. In total, the training set contains 12.5 hours of data while the test set includes 1.25 hours of data.

5.2 Methods in Comparison

To better evaluate the performance of our model, we proposed two other methods for comparison, a baseline model and a LSTM model. The baseline model calculates the average loss rate of the past 5 seconds and then sets the redundancy rate to be multiples of the average loss rate. Then, the ingress gateway will perform RS encoding using this redundancy rate. We also developed a predictive model based on LSTM, inspired by the LSTM model utilized by DeepRS [6]. Rather than 6 packets as a block in [6], our block is much bigger and contains all the packets from a certain frame. The LSTM model takes information from the past 5 seconds and predicts the loss rate of each frame in the next second. Then, it will round the predicted loss rate of each frame to the nearest multiple of 10% as the redundancy rate.

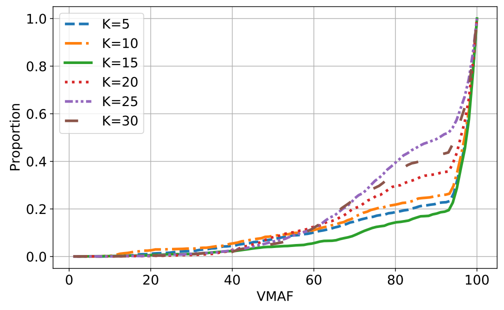


Figure 9: Evaluation results of the RL-AFEC model with different number of critical frames.

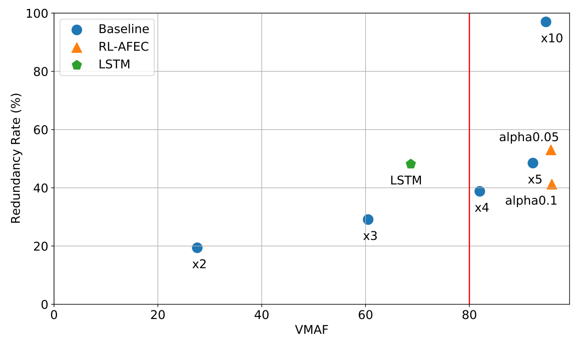
5.3 Number of Critical Frames

For RL-AFEC, we use the idea of critical frames to reduce the action space. The model only determines $K + 1$ actions where K is the number of critical frames. It is important to determine a proper K value. If K is too large, it may result in a huge action space to explore and leads to poor convergence. If K is too small, it would impair the ability of RL-AFEC to adapt to different network scenarios. Therefore, we trained several models by setting K to be different values ($K = 5$, $K = 10$, ..., $K = 30$). Figure 9 shows the evaluation results on the test set. The lower the curve, the better the performance. The green line at the bottom refers to $K = 15$ with the best performance among all models with different K values. The upper two lines corresponding to $K = 25$ and $K = 30$ lead to unsatisfactory performance since the action space is too large to explore. On the other hand, the performance of RL-AFEC would also be impaired when K is too small. For example, the model with $K = 5$ is not as good as the model with $K = 15$ because it lacks flexibility for selecting distinct redundancy rate combinations to cope with various kinds of loss patterns. From the above observations, we set K equal to 15 in the following experiments.

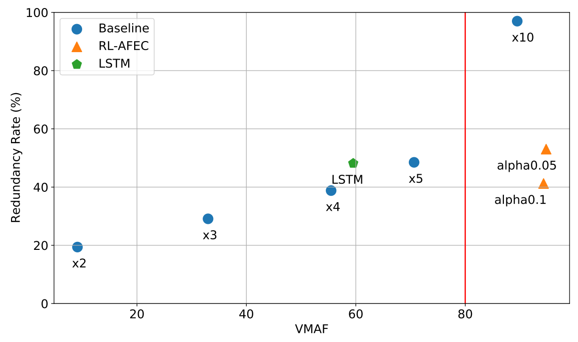
5.4 Evaluation Results on Test Set

In this subsection, we present the evaluation results of the three methods including the RL-AFEC model, the baseline model, and the LSTM model on the test set. For the baseline model, we set the redundancy rate to be several times of the average loss rate of the past 5 seconds, which are denoted as baseline_x2, baseline_x3, baseline_x4, baseline_x5, baseline_x10, respectively. For the RL-AFEC model, we used two α values in Equation 8, $\alpha = 0.05$ and $\alpha = 0.1$. Note that the model with a smaller α tends to consume more bandwidth than that with a larger α.

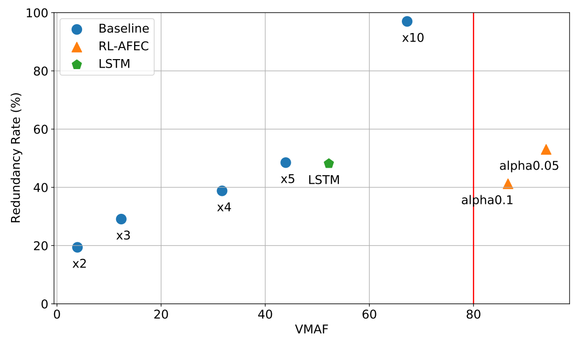
Table 4 summarizes the evaluation results. We provided the mean value and three percentiles in the table (80%, 90%, and 95%). For example, for the RL model with α equal to 0.05, the 80th percentile value is 95.88, which means 80% of tested 1-second videos in the test set have a VMAF score above 95.88. We can compare the performance of different methods more clearly by observing the data points in Figure 10. The vertical red line corresponds to a VMAF score equal to 80, which is the target score used in the reward function and also indicates 'good' video quality according to Netflix. Figure 10(a) shows 80th percentile results. We can observe that the



(a) Evaluation results: 80th percentile



(b) Evaluation results: 90th percentile



(c) Evaluation results: 95th percentile

Figure 10: 80th percentile, 90th percentile, and 95th percentile VMAF score in the test set obtained by different methods. The horizontal axis refers to the VMAF value. The vertical axis is the average redundancy rate consumed by different methods.

performance of baseline_x5 is close to the RL models. However, if we look at the 90th percentile results, the performance of the baseline models and the LSTM model degrades quickly and only the RL-AFEC models can still achieve high VMAF scores with relatively low redundancy rate. To catch up with the performance of RL-AFEC in terms of VMAF, baseline model has to add ten times of the average loss rate as redundancy, which leads to much higher bandwidth consumption compared to RL-AFEC. The 95th percentile results demonstrate more superiority of our methods, since only the RL-AFEC models can maintain high video quality for 95% of test data.

VMAF	Mean	80%	90%	95%
Baseline x2	59.95	27.59	9.13	3.93
Baseline x5	90.37	92.24	70.65	43.94
Baseline x10	93.73	94.75	89.50	67.26
LSTM	84.71	68.72	59.53	52.20
RL(α = 0.1)	95.72	95.71	94.32	86.63
RL(α = 0.05)	97.25	95.88	94.79	93.95

Table 4: Evaluation results of different methods. 80% means there are 80% of tested 1-sec videos whose VMAF is higher than the value in the table.

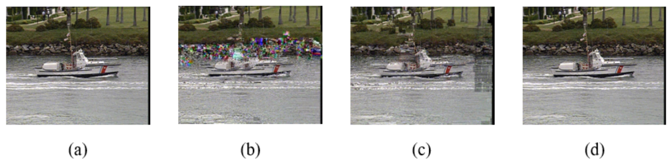


Figure 11: Comparison of subjective video quality of the *coastguard* sequence recovered by the three methods from 7% packet loss rate. (a) Original frame. (b) Baseline x4. (c) LSTM model. (d) RL (α = 0.1).

5.5 Evaluation Results on Different Videos

We also performed evaluation by adding various loss rates to different videos and use the methods in comparison to recover the lost packets. Here we evaluated the RL-AFEC model with α equal to 0.1, the baseline model with redundancy rate equal to 4 times of the average loss rate of past 5 seconds, and the LSTM model. These three methods consumed similar redundancy rate according to previous evaluation results.

The first example is the coastguard sequence with a 7% packet loss rate. From Figure 11, the first picture shows the original video frame without any loss. The next three pictures present the recovered video frame using baseline x4, the LSTM model, and the RL-AFEC model with α equal to 0.1. Obviously, the video recovered using RL-AFEC model has the highest quality, which is almost as good as the original video. But for the other two methods, there is a lot of noise in the picture since they are incapable of recovering lost packets.

The second example is the mother&daughter sequence with a 6% packet loss rate. Figure 12 shows that there are distortions on the people's faces in the two video frames in the middle while the picture recovered by the RL-AFEC model is as perfect as the original. As for the last example shown in Figure 13, we added a 4% loss rate to the news video sequence. We can see that there are still some ribbons on the picture that are caused by packet loss even though the video is recovered by the baseline model and the LSTM model. On the contrary, there is no distortion in the video recovered by the RL-AFEC model.

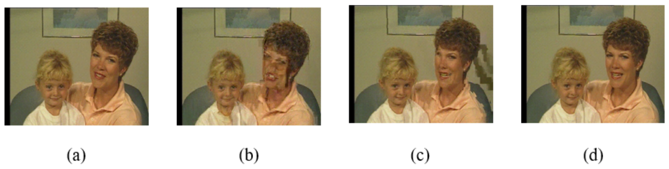


Figure 12: Comparison of subjective video quality of the *mother&daughter* sequence recovered by the three methods from 6% packet loss rate. (a) Original frame. (b) Baseline x4. (c) LSTM model. (d) RL ($\alpha = 0.1$).

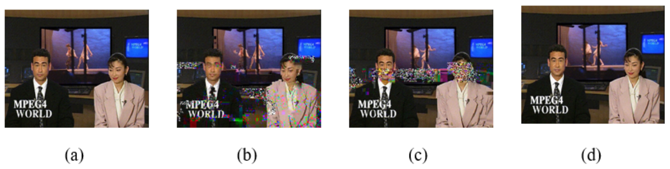


Figure 13: Comparison of subjective video quality of the *news* sequence recovered by the three methods from 4% packet loss rate. (a) Original frame. (b) Baseline x4. (c) LSTM model. (d) RL ($\alpha = 0.1$).

6 CONCLUSION

With an objective of maintaining the quality of videos transmitted over unreliable networks with low bandwidth consumption in FEC, we propose RL-AFEC, an adaptive FEC scheme that learns to select proper redundancy rates for all video frames in each Group of Pictures (GoP) using reinforcement learning, without any domain-specific predefined rules. Then, the redundant packets of each frame are added by frame-level RS code according to the selected redundancy rate. The superiority of our method can be summarized as below:

- **Real time:** In every second, RL-AFEC can select the proper redundancy rate based on the feedback of the past 5 seconds to tackle potential packet loss in the next 1-second interval video transmission.
- **Robust:** According to the evaluation results, for more than 95% 1-second videos in the test set, RL-AFEC achieves a VMAF score larger than 80 (indicating good video quality) with only 40% additional bandwidth consumption.
- **Fast:** RL-AFEC takes less than 5ms on average to select redundancy rates for each frame on ordinary computers, which could be even faster if performed on the SD-WAN gateways.

Yet, there are still areas that may be worth exploring to improve RL-AFEC:

- **Real-world traces:** The evaluations that have been done in this work are based on simulation environments in the laboratory. For example, we introduce time-varying packet loss rates according to the modified GE channel. In the future, we would collect real-world traces, if achievable, to supplement our training dataset and retrain the RL model.
- **Periodically retraining:** In this paper, the RL-based adaptive FEC scheme is trained as an offline task. In other words, once training is done, RL-AFEC remains unmodified. As a result, the performance of the model depends on whether the training set meets all the potential situations in the network. However, we may periodically retrain the model by accommodating future unseen video types and loss patterns into the training set. Going forward, we may deploy the model in a real network and perform online learning, such that RL-AFEC can further adapt itself to the dynamic conditions in the network.

ACKNOWLEDGMENTS

This work was partially supported by Fortinet, Inc., CA. The authors would like to thank Fortinet engineers for providing insightful information through numerous meetings and communications.

REFERENCES

[1] 2019. Preparing Your IP Network for Video Conferencing. https://support.polycom.com/content/dam/polycom-support/products/uc-infrastructure-support/management-scheduling/dma/other-documents/en/preparing-ip-network-video-conferencing.pdf
[2] Enrico Baccaglini, Tammam Tillo, and Gabriella Olmo. 2008. Slice sorting for unequal loss protection of video streams. *IEEE Signal Processing Letters* 15 (2008), 581–584.
[3] BBC. 2021. Facebook remote working plan extended to all staff for long term. Retrieved January 13, 2022 from https://www.bbc.com/news/technology-57425636
[4] Nicholas Bloom. 2020. Stanford research provides a snapshot of a new working-from-home economy. Retrieved January 13, 2022 from https://news.stanford.edu/2020/06/29/snapshot-new-working-home-economy
[5] J-C Bolot, Sacha Fosse-Parisis, and Don Towsley. 1999. Adaptive FEC-based error control for Internet telephony. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, Vol. 3. IEEE, 1453–1460.
[6] Sheng Cheng, Han Hu, Xinggong Zhang, and Zongming Guo. 2020. Deeprs: Deep-learning based network-adaptive fec for real-time video communications. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
[7] Qin Dai and Ralf Lehnert. 2010. Impact of packet loss on the perceived video quality. In *2010 2nd International Conference on Evolving Internet*. IEEE, 206–209.
[8] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679* (2015).
[9] Edwin O Elliott. 1963. Estimates of error rates for codes on burst-noise channels. *The Bell System Technical Journal* 42, 5 (1963), 1977–1997.
[10] Salma Shukry Emara, Silas Fong, Baochun Li, Ashish Khisti, Wai-Tian Tan, Xiaoqing Zhu, and John Apostolopoulos. 2021. Low-latency network-adaptive error control for interactive streaming. *IEEE Transactions on Multimedia* (2021).
[11] Edgar N Gilbert. 1960. Capacity of a burst-noise channel. *Bell system technical journal* 39, 5 (1960), 1253–1265.
[12] Sheila S Hemami and Amy R Reibman. 2010. No-reference image and video quality estimation: Applications and human-motivated design. *Signal processing: Image communication* 25, 7 (2010), 469–481.
[13] Te-Yuan Huang, Polly Huang, Kuan-Ta Chen, and Po-Jung Wang. 2010. Could Skype be more satisfying? A QoE-centric study of the FEC mechanism in an Internet-scale VoIP system. *IEEE Network* 24, 2 (2010), 42–48.
[14] Mushahid Hussain and Abdul Hameed. 2018. Adaptive video-aware forward error correction code allocation for reliable video transmission. *Signal, Image and Video Processing* 12, 1 (2018), 161–169.
[15] Cisco Global Cloud Index. 2018. Forecast and methodology, 2016–2021 white paper. *Updated: February* 1 (2018).
[16] Wenyu Jiang and Henning Schulzrinne. 2000. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *Proc. NOSSDAV*.
[17] Heather Kelly. 2020. Twitter employees don't ever have to go back to the office (unless they want to). Retrieved January 13, 2022 from https://www.washingtonpost.com/technology/2020/05/12/twitter-work-home
[18] Eymen Kurdoglu, Yong Liu, and Yao Wang. 2017. Perceptual quality maximization for video calls with packet losses by optimizing FEC, frame rate, and quantization. *IEEE Transactions on Multimedia* 20, 7 (2017), 1876–1887.

[19] Zhi Li, Christos Bampis, Julie Novak, Anne Aaron, Kyle Swanson, Anush Moorthy, and JD Cock. 2018. VMAF: The journey continues. *Netflix Technology Blog* 25 (2018).

[20] Zhi Li, Christos Bampis, Julie Novak, Anne Aaron, Kyle Swanson, Anush Moorthy, and JD Cock. 2018. VMAF: The journey continues. *Netflix Technology Blog* 25 (2018).

[21] Qiyong Liu, Zhaofeng Jia, Kai Jin, Jing Wu, and Huipin Zhang. 2019. Error resilience for interactive real-time multimedia application. US Patent 10,348,454.

[22] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 197–210.

[23] Anish Mittal, Anush K Moorthy, and Alan C Bovik. 2011. Blind/referenceless image spatial quality evaluator. In *2011 conference record of the forty fifth asilomar conference on signals, systems and computers (ASILOMAR)*. IEEE, 723–727.

[24] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. 2012. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters* 20, 3 (2012), 209–212.

[25] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (2013). http://arxiv.org/abs/1312.5602 NIPS Deep Learning Workshop 2013.

[27] Chinmay Padhye, Kenneth J Christensen, and Wilfrido Moreno. 2000. A new adaptive FEC loss control algorithm for voice over IP applications. In *Conference Proceedings of the 2000 IEEE International Performance, Computing, and Communications Conference (Cat. No. 00CH37086)*. IEEE, 307–313.

[28] Lawrence Rabiner and Biinghwang Juang. 1986. An introduction to hidden Markov models. *ieee assp magazine* 3, 1 (1986), 4–16.

[29] S Rajagopalan. 2020. An Overview of SD-WAN Load Balancing for WAN Connections. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 1–4.

[30] Ashwin Rao, Arnaud Legout, Yeon-sup Lim, Don Towsley, Chadi Barakat, and Walid Dabbous. 2011. Network characteristics of video streaming traffic. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. 1–12.

[31] Irving S Reed and Gustave Solomon. 1960. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics* 8, 2 (1960), 300–304.

[32] P. Seeling and M. Reisslein. 2012. Video Transport Evaluation With H.264 Video Traces. *IEEE Communications Surveys and Tutorials, in print* 14, 4 (2012), 1142–1165. Traces available at `trace.eas.asu.edu`.

[33] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[34] Suramya Tomar. 2006. Converting video formats with FFmpeg. *Linux Journal* 2006, 146 (2006), 10.

[35] Tomoaki Tsugawa, Norihito Fujita, Takayuki Hama, Hideyuki Shimonishi, and Tutomu Murase. 2007. TCP-AFEC: An adaptive FEC code control for end-to-end bandwidth guarantee. In *Packet Video 2007*. IEEE, 294–301.

[36] Thierry Turletti and Christian Huitema. 1996. Videoconferencing on the Internet. *IEEE/ACM Transactions on networking* 4, 3 (1996), 340–351.

[37] Michael Wood. 2017. How to make SD-WAN secure. *Network Security* 2017, 1 (2017), 12–14.

[38] Huahui Wu, Mark Claypool, and Robert Kinicki. 2005. Adjusting forward error correction with temporal scaling for TCP-friendly streaming MPEG. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 1, 4 (2005), 315–337.

[39] Jiyan Wu, Bo Cheng, Ming Wang, and Junliang Chen. 2016. Priority-aware FEC coding for high-definition mobile video delivery using TCP. *IEEE Transactions on Mobile Computing* 16, 4 (2016), 1090–1106.

[40] Jiyan Wu, Chau Yuen, and Junliang Chen. 2015. Leveraging the delay-friendliness of TCP with FEC coding in real-time video communication. *IEEE Transactions on Communications* 63, 10 (2015), 3584–3599.

[41] Jimin Xiao, Tammam Tillo, Chunyu Lin, and Yao Zhao. 2012. Dynamic sub-GOP forward error correction code for real-time video applications. *IEEE Transactions on Multimedia* 14, 4 (2012), 1298–1308.

[42] Jimin Xiao, Tammam Tillo, and Yao Zhao. 2013. Real-time video streaming using randomized expanding Reed–Solomon code. *IEEE transactions on circuits and systems for video technology* 23, 11 (2013), 1825–1836.

[43] XK Yang, Ce Zhu, ZG Li, Xiao Lin, GN Feng, Si Wu, and Nam Ling. 2003. Unequal loss protection for robust transmission of motion compensated video over the internet. *Signal Processing: Image Communication* 18, 3 (2003), 157–167.

[44] Zhenjie Yang, Yong Cui, Baochun Li, Yadong Liu, and Yi Xu. 2019. Software-defined wide area network (SD-WAN): Architecture, advances and opportunities. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–9.

[45] Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. 2020. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 6672–6679.

[46] Minghao Ye, Junjie Zhang, Zehua Guo, and H. Jonathan Chao. 2021. DATE: Disturbance-Aware Traffic Engineering with Reinforcement Learning in Software-Defined Networks. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. 1–10. https://doi.org/10.1109/IWQOS52092.2021.9521343

[47] Junjie Zhang, Minghao Ye, Zehua Guo, Chen-Yu Yen, and H Jonathan Chao. 2020. CFR-RL: Traffic engineering with reinforcement learning in SDN. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2249–2259.

A ARTIFACT APPENDIX

A.1 Abstract

This appendix serves to provide information for reproducibility of the RL-AFEC. Please find more detailed information in the corresponding GitHub repository: https://github.com/chenke97/RL-AFEC

A.2 Artifact check-list (meta-information)

- **Algorithm: Reinforcement learning**
- **Data set: RTP packet stream generated by simulation**
- **Run-time environment: Python, Tensorflow**
- **Hardware: Powerful CPU with multiple cores for training is recommended**
- **Metrics: PL-VQA**
- **Output: FEC redundant rate for each video frame.**
- **Experiments: Testing on the test set and different per-video streams.**
- **How much disk space required (approximately)?: < 1 GB**
- **How much time is needed to prepare workflow (approximately)?: < 1 hour**
- **How much time is needed to complete experiments (approximately)?: 12 hours**
- **Publicly available?: Yes**
- **Code licenses (if publicly available)?: MIT license**
- **Data licenses (if publicly available)?: MIT license**
- **Archived (provide DOI)?: https://doi.org/10.1145/3524273.3528184**

A.3 Description

A.3.1 How delivered. For training, we used NYU HPC server to accelerate policy convergence. Basically, we used 20 CPU cores, 19 of them serve as worker threads which interact with environment (e.g. 1. obtain a state 2. pick up an action 3. get an reward in terms of video quality score generated by PL-VQA model in our paper) to collect experience and 1 central thread performs back-propagation to update weights of the neural network. The total training time is around 8 hours.

For testing, local laptop's computing power is enough. We ran our testing on an Intel CORE i7 CPU. The inference time for one data point (the state for 1-second video) is around 2 ms.

A.3.2 Hardware dependencies. A powerful CPU is recommended to train the agent. It relies on sufficient interaction with environment through collecting experiences by multiple threads running at the same time to train a good policy; hence, a powerful CPU with multiple core may accelerate the whole process.

A.3.3 Software dependencies. Python 3.6.8, Tensorflow 2.2.0 (CPU version), numpy, tqdm, absl

A.3.4 Data sets. We generated both training set and test set in simulation. Data sets are available on GitHub.

A.4 Installation

We used Anaconda to create an virtual environment rl-afec and then install required dependencies.
1. conda create -n rl-afec python=3.6
2. conda activate rl-afec
3. pip install tensorflow-cpu==2.2.0
4. pip install numpy tqdm absl-py

A.5 Experiment workflow

1. Training: To train an RL-AFEC agent, put the trace file (e.g., video_RL_train.txt) in data/, then specify the file name in config.py, i.e., trace_file = 'video_RL_train.txt'. Then run "python3 train.py". The saved models will be stored in folder "tfckpt/RLRS_1.0pure_policy_video_RL_train.txt/" (folder name format is "RLRS_1.0pure_policy_" + training set file name).

2. Testing To test the trained policy on a set of test traces, put the test trace file (e.g., video_RL_testing.txt) in data, then specify the file name in config.py, i.e., test_trace_file = 'video_RL_testing.txt'. The latest checkpoint will be loaded by default.

A.6 Evaluation and expected result

The testing results will be saved in file "eval_results_video_RL_testing.txt" and "loss_pattern_after_video_RL_testing.txt" (file name format is "eval_results_" + test set file name). In eval_results_video_RL_testing.txt, each row will contain selected actions, VMAF score after recovery and bandwidth wasted (in byte) for 30 frames of every second. For example, for a row "[6, 5, 4, 3, 2, 3, 2, 5, 5, 3, 5, 6, 6, 3, 5, 2] 96.01322937011719 567000", the first part are the selected actions for each of 15 critical frames and the shared redundancy rate for the rest non-critical frames, the middle part 96.01 is the video quality score of this one-second video after FEC recovery using the selected actions and the last part 567KB is the wasted bandwidth. The actions above are actually the index into the real action pool [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1], each number represents a redundant rate for performing frame-level Reed-Solomon code. According to the action index, the corresponding redundancy is selected.