Vesper: Learning to Manage Uncertainty in Video Streaming

Bo Chen[1], Mingyuan Wu[1], Hongpeng Guo[1], Zhisheng Yan[2], Klara Nahrstedt[1]
[1]University of Illinois at Urbana-Champaign, [2]George Mason University

ABSTRACT

Video codecs are crucial in video streaming systems. However, the quantization operation in existing codecs introduces irreversible jitters. Moreover, the common practice of fitting a single codec to diverse video content lacks the flexibility to adapt the parameters of a codec for specific content. They lead to the problem of quantization and content uncertainty. Our preliminary study shows an ideal codec without uncertainty gains a significant advantage over the conventional codec with uncertainty. However, realizing the ideal codec presents tremendous challenges in the generalizability and the costs of computation, transmission, and delay. In this paper, we present Vesper, a video streaming system that innovatively tackles uncertainty with two learning-based components, *super-precision* and *self-evolution*. The super-precision module builds a neural network that predicts original feature values from quantized feature values, which effectively mitigates the impact of quantization without inducing generalizability issues. The self-evolution module performs content-aware adaptation on the encoder and replaces non-content-aware video segments with content-aware ones on the fly, which addresses content uncertainty without adding significant costs to on-demand streaming. Evaluations demonstrate Vesper's superior Quality of Experience compared to streaming systems built with state-of-the-art codecs.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Video Streaming, Video Compression, Machine Learning

ACM Reference Format:
Bo Chen[1], Mingyuan Wu[1], Hongpeng Guo[1], Zhisheng Yan[2], Klara Nahrstedt[1]. 2024. Vesper: Learning to Manage Uncertainty in Video Streaming. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3625468.3647621

1 INTRODUCTION

Video streaming is a popular and dominant form of content delivery. According to Sandvine's 2023 Global Internet Phenomena Report [47], video streaming accounts for 65% of all internet traffic. The video codec plays a key role in video streaming by ensuring bandwidth-efficient and high-quality transmission of the video. Traditional video codecs like H.264 [54] and H.265 [50] have achieved great success and are widely adopted in industry platforms including Netflix [24], Hulu [8] and HBO Max [9]. With the advancement in deep learning [30], researchers have built learned video codecs from deep neural networks [2, 37, 44, 60], which can dramatically outperform traditional codecs.

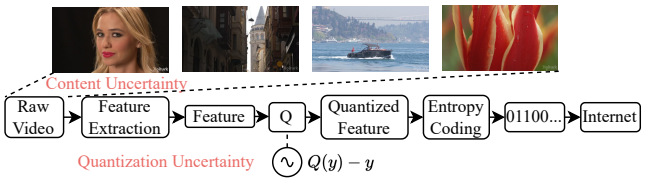


Figure 1: The uncertainty in video streaming stems from the video codec due to the jitter caused by quantization (Q) and the diverse unseen video content.

Despite the success of traditional and learned video codecs, both referred to as "conventional codecs", they struggle to effectively handle the uncertainty present in video streaming, which hampers their *coding efficiency*, i.e., the capability to represent a video with high quality and low bitrates. This uncertainty encompasses quantization and video content, as depicted in Figure 1.

- *Quantization uncertainty* results from the quantization operation in video encoding. Quantization maps continuous values to a finite set of discrete values, which are necessary for entropy coding [22, 55] and network delivery. However, the `round()` operation, widely existing in quantization, maps values in a range uniformly to a single value. This operation alters the video features generated in encoding. The change of feature values is termed the *jitter*. For instance, the `round()` operation in Python maps feature values from the range of $[-0.5, 0.5]$ to zeros, which is equivalent to a jitter within $[-0.5, 0.5]$.
- *Content uncertainty* arises from the common practice that constructs a codec to fit all types of video content, such as travel, gaming, tutorial, and vlog videos [14]. Although this practice offers great generalizability in video codecs, it lacks the flexibility to adapt parameters in the codec to better fit a specific type of content, i.e., *content-aware adaptation*.

To quantify the impact of uncertainty, we conduct preliminary studies that contrast a conventional codec where quantization or content uncertainty exists, and an "ideal" codec, whose uncertainty is removed hypothetically. This study is conducted on a pre-trained learned codec, which demonstrates an ideal codec without quantization uncertainty reduces the training loss of the conventional codec by 29%, and the ideal codec without content uncertainty achieves a 49% decrease in bitrate while maintaining superior video quality. These results quantitatively reveal the sub-optimality of conventional codecs.

MMSys '24, April 15–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0412-3/24/04…$15.00
https://doi.org/10.1145/3625468.3647621

While the preliminary study shows promising results of the ideal codec without uncertainty, realistically implementing the ideal codec poses non-trivial challenges. First, the derivation of the ideal codec is equivalent to overfitting the conventional codec to a specific case of quantization jitter, zero jitter. As a cost, the ideal codec loses generalizability to other cases of jitters, making it impractical for random quantization jitters in realistic cases. Second, deriving an ideal codec with content-aware adaptation involves computation costs to train the codec and transmission costs to deploy the trained codec on the distributed encoder and decoder. Additionally, content-aware adaptation takes a substantial amount of time in training, which is likely to delay video streaming.

We present Vesper, a novel on-demand video streaming system that tackles uncertainty with two key components: super-precision and self-evolution. The *super-precision* module involves a prediction network to predict original feature values from quantized feature values at the video decoder, a prediction loss to bridge the gap between the predicted and original feature values, and a unique training procedure optimizes the prediction network based on the prediction loss. Such a design allows the video decoder to approximate the original feature values more closely than quantized feature values, which mitigates the impact of quantization. As the codec is trained with quantization, it maintains tolerance to quantization jitters. The *self-evolution* module performs the content-aware adaptation of the encoder to improve coding efficiency on specific video contents. It also applies on-the-fly segment replacement to upgrade old segments with those produced by content-aware encoders at the media server. As the self-evolution module involves only the encoder, it runs on the media server, which requires no extra computation at the end viewer and no transmission cost. In addition, by running on-the-fly segment replacement, the on-demand serving process of video segments experiences no extra delay.

Vesper outperforms video streaming systems with state-of-the-art traditional and learned codecs in our evaluations. It improves the average QoE by up to 9% and 31% compared to best-performing learned and traditional approaches, respectively. In our case study, Vesper showcases the ability to effectively reduce the bitrate of a video by up to 45%, while simultaneously improving its quality by 1.4 dB in Peak Signal-to-Noise Ratio (PSNR).

We summarize our contribution as follows.

1) We introduce Vesper, a novel video streaming system that tackles uncertainty with learning.
2) We design a super-precision module to tackle quantization uncertainty via the co-design of the prediction network and prediction loss.
3) We develop a self-evolution module to address content uncertainty with content-aware encoder adaptation and on-the-fly segment replacement.
4) We conduct comprehensive evaluations of Vesper and demonstrate significant improvement compared to existing video streaming systems.

2 BACKGROUND AND MOTIVATION

2.1 Background

Video streaming. As depicted in Figure 2, an on-demand video streaming system consists of two main components: the media server and the end viewer. On the server, raw frames from the video source are encoded offline into fixed-duration video segments. Typically, multiple versions of the same video segment are encoded at different bitrates and stored in the storage, ready for Dynamic Adaptive Streaming over HTTP (DASH). The end viewer utilizes an adaptive bitrate (ABR) algorithm to request and download encoded segments of suitable bitrates from the video streamer. Once a segment is downloaded, it undergoes decoding to convert it from bits into viewable frames. These reconstructed frames are then cached in the buffer and presented to the viewer at an ideal real-time rate, such as 30 frames per second (fps).

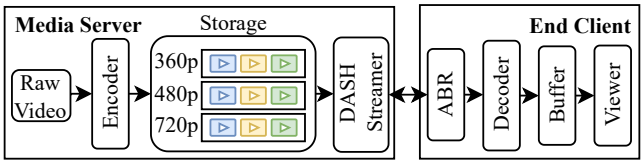


Figure 2: Canonical on-demand video streaming pipeline.

Metric. The primary objective of a video streaming system is to provide users with a high-quality and seamless viewing experience. Therefore, researchers have focused on modeling QoE in video streaming, considering two main factors: video quality (q) and rebuffering (r) [48]. Video quality is determined by the difference between raw frames and reconstructed frames, typically measured using the PSNR metric. Rebuffering is quantified by the rebuffer rate, which represents the ratio of rebuffer duration to the total streaming session duration. The QoE metric can be defined mathematically as shown in Equation 1.

$$QoE = q - \gamma r, \tag{1}$$

where γ is a parameter balancing q and r. The value of γ can be derived based on the size and quality of video segments and the max/min buffer levels according to BOLA [48].

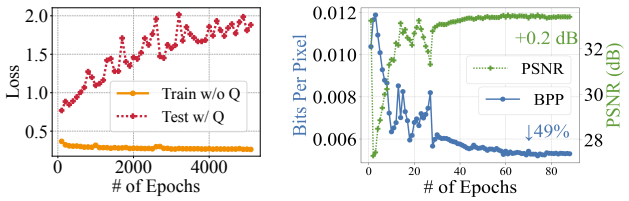
Video codec. The video codec, which includes both the encoder and decoder components in Figure 2, plays a crucial role in achieving a high QoE. In its raw form, a video, such as RGB format, requires 24 bits to represent each pixel. However, modern video codecs have the capability to encode the video using less than one bit per pixel while maintaining a satisfactory level of video quality [50, 54]. This ability to represent a video with high quality and low bit/bandwidth consumption is referred to as *coding efficiency*.

The high coding efficiency of modern codecs, such as x264 [54] and x265 [50], enables smooth video streaming with high quality. With low bandwidth consumption, the likelihood of rebuffering is reduced, resulting in a seamless viewing experience. As deep learning techniques have emerged, researchers have started exploring the representation of video codecs using deep neural networks, such as SSF [2] and ELFVC [44]. These neural network-based codecs have shown better coding efficiency compared to traditional codecs.

2.2 Intuitive Understanding of Limitations

Despite the success of existing video codecs [2, 44, 50, 54], they fail to effectively handle the uncertainty inherent in video streaming. This uncertainty can be categorized into two main types:

1) Quantization uncertainty: Quantization is a crucial step in video codecs that discretizes the feature values. The uncertainty stems from the fact that quantization uniformly maps continuous values in a range to a single discrete value, causing irreversible changes to the feature value, termed jitters. For instance, the quantization of feature values from the range of $[-0.5, 0.5]$ to zero introduces jitters in the range of $[-0.5, 0.5]$. Such jitter exists in the quantization of both traditional video codecs and learned video codecs. More importantly, quantization irreversibly alters the feature values and affects the coding efficiency.
2) Content uncertainty: Video streaming systems need to accommodate diverse content types, which exhibit variations in color distribution and spatial-temporal correlation. For example, travel videos may have vibrant colors, while gaming videos could have darker tones. Additionally, the temporal correlation may be high in slow-motion tutorial videos but low in fast-motion adventure vlog videos. However, existing video codecs adopt the common practice of constructing a codec to fit all possible contents, without content-aware adaptation. Although this practice offers good generalizability, it constrains the coding efficiency that could be gained from content-aware adaptation.



(a) An ideal codec without quantization performs 29% better in terms of training loss.

(b) An ideal codec with content-aware adaptation consumes 49% less bandwidth and 0.2 dB better PSNR.

Figure 3: The codec archives better performance when uncertainty is ideally removed.

2.3 Quantitative Study of Limitation

The above description of the limitation is on an intuitive level. To quantitatively understand the above limitation, we compare a conventional codec to an ideal one, whose quantization or content uncertainty is ideally removed. Designing an ideal codec requires adapting its parameters under a different setting, which cannot be trivially achieved with traditional video codecs whose parameters are handcrafted. To this end, we utilize the state-of-the-art learned video codec, ELFVC [44], as the conventional codec in the preliminary study, whose parameters are easily adaptable via back-propagation. The ELFVC codec is pre-trained on the Vimeo-90k dataset [57]. Next, we show how the quantization and content uncertainty are ideally removed and the impact.

Quantization uncertainty. To ideally remove quantization uncertainty, we fine-tune the pre-trained codec on the Vimeo-90k dataset without quantization. In existing approaches [2, 37, 44], the quantization in training is simulated by adding uniform noise to the feature. Therefore, we remove the impact of quantization by skipping the noise addition during training. In Figure 3(a), we illustrate the training loss (orange curve), which evaluates both video reconstruction distortion and bitrate of the learned codec. Every point represents the value of the loss function evaluated at each epoch, representing a complete pass over the Vimeo-90k dataset. The training loss at the beginning (epoch#0) characterizes the performance of the conventional codec. The subsequent epochs indicate how the training loss converges when the codec is gradually fine-tuned without quantization. The result indicates that the removal of quantization reduces the training loss of the conventional codec (epoch#0) by up to 29% after training for a sufficient time.

Content uncertainty. To ideally remove content uncertainty, we arbitrarily select the "Jocky" video from the UVG dataset [40]. Then, we perform content-aware adaptation by fine-tuning the pre-trained codec on this specific video until convergence. Figure 3(b) illustrates the test performance of the codec on the same video in bit consumption (bits per pixel) and quality (PSNR). Epoch#0 denotes the conventional codec and the subsequent epochs represent the ideal codec fine-tuned for different numbers of iterations. Each epoch represents a complete pass over the entire video. The result shows that removing content uncertainty drastically improves the codec by reducing bit consumption by up to 49% and improving PSNR by up to 0.2 dB.

Overall, quantization and content uncertainty noticeably constrain the performance of the codec.

2.4 Challenge, Intuition, and Solution

Bridging the gap between an ideal codec and a realistic one is non-trivial. We describe the challenges involved with quantization and content uncertainty and discuss our intuition and solution for them as follows.

2.4.1 Quantization Uncertainty. An ideal codec trained without quantization is equivalent to the conventional codec overfitted to a special case of quantization jitter, zero jitter. As a result, the ideal codec loses generalizability to other cases of quantization jitters, making it impractical for the realistic scenario. Figure 3(a) shows the result of applying the codec trained without quantization on the UVG dataset [40] with realistic quantization. The test loss (red curve) becomes divergent with more training epochs, which indicates the loss of generalizability in the process of training (overfitting).

Intuition. Pixel values in a video provide information about its content even if they are slightly altered. From a machine learning perspective, it is shown that contents in blurred images can still be classified and detected [34, 49], and the original image can be reconstructed from a noisy image [21]. From a human perspective, if an image is noisy because of the limitation of the capture device or shaking while capturing, it is still possible to identify the object or people in the image. Similarly, quantization can also be treated as an operation that alters feature values with jitters, making it possible to recover the original values even after quantization.

Solution. Based on the intuition above, we comprehensively design a prediction network (§3.1.1) that predicts the original feature values with the input of quantized feature values and a prediction loss (§3.1.2) that models the gap between predicted and original feature values. A unique training procedure (§3.3) minimizes the prediction loss to ensure the predicted feature values approximate the original

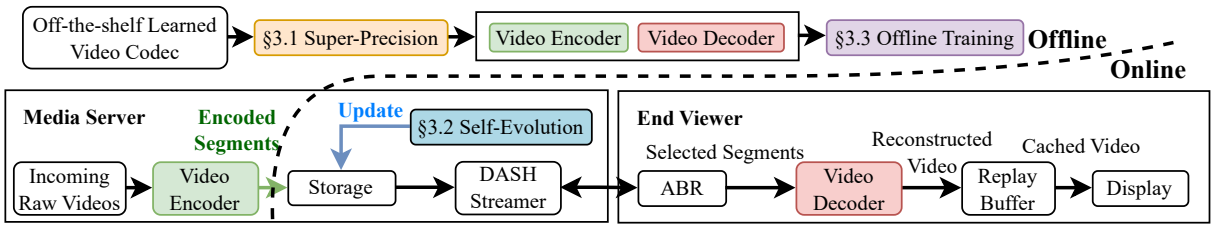


Figure 4: Vesper Overview.

feature values more precisely than the quantized feature values. Hence, we could mitigate the impact of quantization uncertainty without affecting the functionality of the codec.

2.4.2 Content Uncertainty. Content-aware adaptation incurs computation costs due to model training and transmission costs to deploy the trained codec to the distributed video encoder and decoder. Additionally, content-aware adaptation inevitably delays the production of video segments, which potentially increases the delay of video streaming.

Intuition. Learned codecs derived via back-propagation are flexible. Such flexibility allows the codec to benefit from training even with a portion of its parameters fixed during optimization [26]. This observation allows us to selectively update parameters, avoiding certain computation and transmission costs. Moreover, in on-demand streaming, the production and serving of video segments are asynchronous. When the serving proceeds with video segments produced by a non-content-aware codec, we can replace the old video segments with optimized ones without introducing any delay.

Solution. We leverage the flexibility of the codec to perform content-aware encoder adaptation (§3.2.1), which optimizes parameters in the encoder at the media server. Therefore, it requires no computation cost on the end viewer or server-viewer transmission cost. Based on the asynchronous characteristic of on-demand streaming, we apply on-the-fly segment replacement (§3.2.2) that replaces old segments with those produced by content-aware encoders, which seamlessly improves coding efficiency for on-demand streaming.

3 VESPER DESIGN

Vesper consists of an offline stage and an online stage. In the offline stage, the *super-precision* module (§3.1) transforms the base codec to better handle quantization uncertainty based on the design of the prediction network and the prediction loss. Then, the video encoder and decoder are offline trained (§3.3) and deployed to the media server and the end viewer, respectively. After that, the video encoder encodes the raw video into segments of varying bitrates and saves it in storage for DASH streaming.

In the online stage, the video segments are fetched by the end viewer based on the ABR algorithm from the DASH streamer. The downloaded segments are decoded and cached in the buffer before being displayed. Meanwhile, the *self-evolution* module (§3.2) performs content-aware optimization on the encoder, which replaces old video segments with those produced by content-aware encoders on the fly.

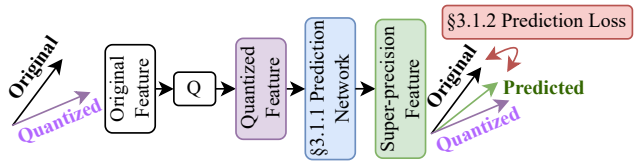


Figure 5: The super-precision module utilizes a prediction network to derive the super-precision feature from quantized values. The prediction loss is minimized to ensure the super-precision feature well approximates the original feature. This process is equivalent to regaining the "lost precision" in quantization.

3.1 Super-Precision

Figure 5 describes the workflow of the super-precision module. After the original feature is quantized, a *prediction network* (§3.1.1) generates a *super-precision* feature from the quantized feature. The distance between the super-precision feature and the original feature is quantified by the *prediction loss* (§3.1.2). The prediction loss will be minimized in training (§3.3) to ensure the super-precision feature is a better approximation of the original feature (than the quantized feature).

3.1.1 Prediction Network. Prediction of the original feature values from quantized values resembles image denoising. One close effort is Model Diffusion [21], which has recently demonstrated its strength in image denoising and holds the potential for mitigating feature jitters induced by quantization. The approach is to apply U-Net [45] several times to lower the noise in the image iteratively [21]. Here, U-Net is a neural network design constructed by convolution layers, residual networks [20], and attention modules [52].

Challenge. Nevertheless, this approach usually requires applying U-Net up to 1000 times [21]. It makes the video decoding computation-intensive and real-time video streaming impractical.

Single-pass denoising. In a recent study, it is shown that the first pass of U-Net brings substantially more improvement in image coding efficiency than subsequent passes [38]. Leveraging this observation, it is reasonable to apply U-Net [45] once to denoise the quantized feature, which remains effective without introducing too much computation overhead. Based on this intuition, we denote the prediction network as Q^R, which takes the quantized feature as input and outputs the predicted feature. The predicted feature, termed *super-precision* feature $\tilde{y}_t$, serves as a more precise approximation of the original feature than the quantized feature. This process is formulated in Equation 2.

$$\tilde{y}_t = Q^R(Q(y_t)), \tag{2}$$

Here, Q represents the quantization operation, while y_t corresponds to the quantized feature.

3.1.2 Prediction Loss. To optimize the prediction network, a crucial step is to devise an appropriate loss function that bridges the gap between the original feature and the super-precision feature.

Challenge. While uniformly suppressing the difference between the original feature and the super-precision feature with mean square error (MSE) is straightforward, we find it marginally improves or even hampers coding efficiency. The reason is potentially minimizing the MSE of features is not equivalent to maximizing the coding efficiency.

L2-norm-based loss. Seeking a loss function that could lead to better coding efficiency, we narrow down our options to four commonly used functions to measure distance: the L1 norm [28], the L2 norm [41], the Smooth L1 norm [18], and the cosine similarity [46]. Empirically, we utilize these terms to measure the difference between the original feature and the super-precision feature in the codec. Our empirical studies indicate that the L2 norm yields the best overall performance. Consequently, we incorporate an L2 norm-based prediction loss term in training (detailed in Equation 4).

3.2 Self-Evolution

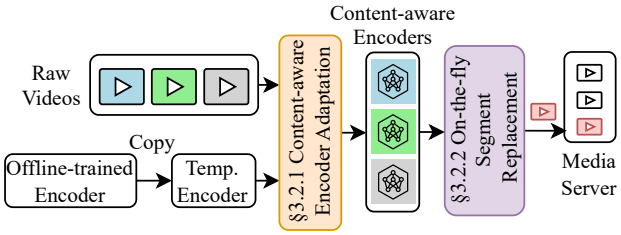


Figure 6: The self-evolution module first optimizes the encoder for every target video with back-propagation. Then, it replaces old segments in the media server on the fly. Coding efficiency continuously improves for all videos in on-demand serving.

Figure 6 presents an overview of self-evolution. Given a raw video, it optimizes a temporary encoder, copied from the offline-trained video encoder (§3.2.1). This step produces content-aware encoders per video. After that, the content-aware encoder produces video segments with optimized coding efficiency, which will update the old segments in the media server on the fly (§3.2.2).

3.2.1 Content-aware Encoder Adaptation. To avoid the impact on the video decoder, we freeze parameters in the video encoder that are shared by the video decoder. By minimizing the loss function in §4, we update non-sharable parameters, i.e., the motion encoder and the residual encoder. As such, we ensure no communication and viewer-side computation is required for self-evolution.

Challenge. After selecting parameters to update, the natural approach is to simultaneously optimize all non-freeze parameters. However, we observed that this approach results in minimal or negative improvements and is slow. The reason is likely that the prediction network is susceptible to the statistical distribution of the original features, as described in §3.3. Such a property affects the optimality and convergence of the model. Nevertheless, the greedy training strategy like §3.3 does not apply to content-aware encoder adaptation where the whole network architecture is fixed.

Progressive optimization. To tackle this challenge, the key observation is that, in contrast to optimizing all parameters simultaneously, optimizing parameters either before or after quantization can better improve the training loss. Based on this observation, we device content-aware encoder adaptation to optimize non-freeze parameters progressively. Specifically, we first optimize parameters in the motion encoder until convergence. Then, we continue with parameters in the residual encoder.

3.2.2 On-the-fly Segment Replacement. To replace the video segments, it is straightforward to optimize the codec until convergence and then update the old segments with ones produced by the content-aware encoder.

Challenge. However, the benefits of self-evolution cannot be enjoyed by the viewers until the optimization converges, which requires iterating the video tens of times.

Immediate storage update. To overcome this downside, we choose to perform an immediate storage update. Specifically, the temporary encoder's coding efficiency is continuously tracked after a complete pass of a video. Whenever the encoder's training performance is improved, we produce the video segments with the adapted encoder and update the corresponding segments in the storage. This technique ensures the benefits of content-aware encoder adaptation can be swiftly reflected on the viewer side before self-evolution is completed.

3.3 Offline Training

With the expression in Equation 2, gradients in the network cannot be properly back-propagated as the quantization operation Q is non-differentiable. Inspired by the straight-through estimator [6], we rewrite the super-precision feature for training, as indicated in Equation 3.

$$\tilde{y}_t = d(\tilde{y}_t - y_t) + y_t, \tag{3}$$

where $d(x)$ represents the "detach" function of PyTorch [42] that stops calculating gradients for x.

Challenge. To train a learned video codec with super-precision, the initial thought is to train the entire network in an end-to-end manner. However, the training can hardly converge with this approach. We speculate the reason is that the prediction network, unlike traditional tasks, e.g., classification, detection, and denoising, whose input is continuous, takes quantized values as input more susceptible to the statistical distribution of the original features. For instance, an input value of 1 should probably yield the output of 1.4 in one distribution but 0.6 in another, which makes a huge difference.

Such susceptibility makes training unstable. Specifically, a slight change in the parameters of the video encoder would alter the distribution of the original feature values, which might significantly degrade the performance of the prediction network. The degradation of the prediction network would require an update of its parameters, which affects the video encoder again and changes the distribution of original feature values. As a result, a vicious cycle is established, where the prediction network and the codec constantly fail to converge, hindering the overall training process.

Greedy training. To prevent the vicious cycle, we have to stabilize the original feature values when training the prediction network. This can be accomplished by 1) training all modules without the prediction network and 2) freezing the modules responsible for generating the feature values before applying and training all other parameters. Here, freezing a module means keeping its parameters constant while optimizing the parameters of other modules. As such, both the codec and the prediction network can be optimized in a stable way.

Considering that the video codec sequentially generates two types of feature values, motion and residual, the above technique needs to be performed twice. Hence, we propose a greedy training procedure that involves three stages: warmup, motion super-precision (*MSP*), and residual super-precision (*RSP*).

(1) Warmup stage: All codec modules are trained simultaneously without super-precision.
(2) *MSP* stage: The codec components generating the motion features are frozen to ensure stability. The quantized motion features are enhanced with super-precision and all parameters except those of the motion encoder are optimized.
(3) *RSP* stage: The codec modules generating the residual features are frozen. Both the motion and residual features are enhanced with super-precision and all parameters except those of motion encoder, motion decoder, and residual encoder are optimized.

Such a training procedure allows for a successful convergence and improved performance of the codec.

4 IMPLEMENTATION

Codec. To support ABR algorithms that require different bitrates, it is essential to compress videos at different compression levels, denoted as $l = 1, 2, ..., L$. However, the rate-distortion trade-off of many learned video codecs is typically fixed, determined by a training hyper-parameter [2, 37, 60]. To overcome this limitation, our base codec adopts the ELFVC approach [44], where the compression level is treated as a configurable input to the learned codec. The concept involves converting compression levels into one-hot vectors, spatially tiling them, and concatenating them to the input of various neural network modules.

Loss function. The loss function used in super-precision and self-evolution is presented in Equation 4.

$$\mathcal{L} = \sum_{k=1}^{K} \mathop{\mathbb{E}}_{l} [\lambda^l \mathcal{D}(x_k, \hat{x_k}^l) + r(\hat{b}_k^l) + \alpha \sum_{y \in \mathcal{S}} \left\| \tilde{y}_k^l - y_k^l \right\|_2]. \quad (4)$$

In this equation, the variable l (ranging from 1 to 8) represents the compression level, while λ^l determines the trade-off between rate and distortion for different compression levels. The parameter K corresponds to the number of P frames in a Group of Pictures (GOP). The I frame, denoted as x_0, undergoes encoding and decoding using the Better Portable Graphics (BPG) image codec [5]. Following the approach of DVC [37], we set the smallest trade-off parameter to $\lambda^0 = 256$, with subsequent λ values being twice the previous one ($\lambda_l = 256 \times 2^l$). This range of λ values covers a sufficient range of bitrates for ABR streaming. The distortion between the original frame x_k and the frame reconstructed at level l, $\hat{x}_k^l$, is evaluated using the MSE and denoted by $\mathcal{D}(x_k, \hat{x}_k^l)$. The estimated bits per pixel of frame k at level l is represented by $\hat{b}_k^l$. The set $\mathcal{S} = \{y^{(f)}, y^{(r)}\}$ comprises the motion and residual feature values. The variables y_k^l and $\tilde{y}_k^l$ denote the feature values produced at compression level l and frame k before and after quantization, respectively. The hyper-parameter α controls the learning rate of the prediction network.

Offline training configurations. Offline training minimizes the loss function in Equation 4. The levels in each training sample are randomly and iteratively generated like [44]. We employ the Adam optimizer [27] with an initial learning rate of 10^{-4}. The learning rate is reduced by a factor of 10 after convergence until it reaches 10^{-6}. We utilize the Vimeo-90k dataset [57] for the training, consisting of over 70k training samples. Each sample includes seven images, with the first image as the I frame ($k = 0$) and the remaining images as P frames ($K = 6$). The training sequences are randomly cropped and resized to 256×256 resolution with a batch size of $B = 8$ following [44]. Training lasts approximately 10 epochs for warmup, and 15 epochs each for *MSP* and *RSP*, taking around 150 hours on an NVIDIA RTX 3090 Ti GPU. Considering that training is conducted offline and only once, the cost is reasonable.

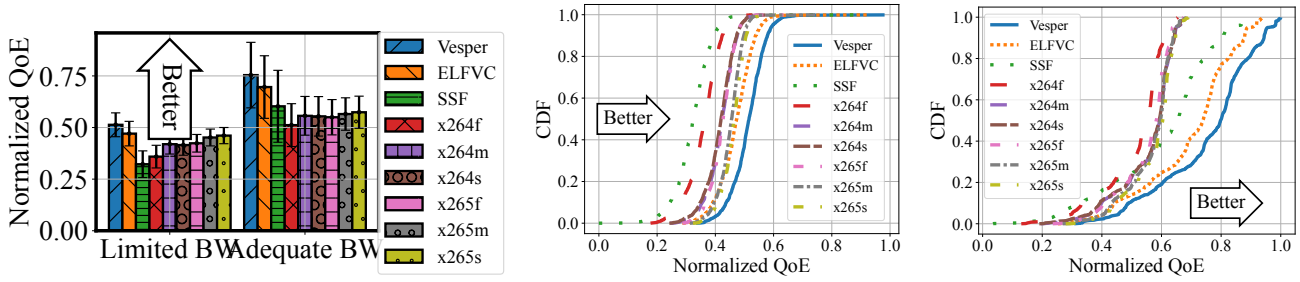
Self-evolution configuration. Self-evolution adopts most of the offline training configuration except that i) the training video is resized to a resolution of 256×256 without random cropping, ii) each training sample is a 16-frame GOP, following [44], ($K = 15$ and $B = 1$), and iii) we report the results obtained with a single-epoch self-evolution unless specified otherwise.

Streaming. In Vesper, we employ BOLA (Buffer-Based Optimized Linear Algorithm) [48], an industrial-level Adaptive Bitrate (ABR) algorithm [12]. BOLA determines the bitrate for each segment download based on the buffer status. The segment size is set to 5 seconds following the approach in [13]. The streaming client is implemented on a Linux desktop equipped with an AMD Ryzen 9 5900X CPU running at 4.95GHz and an NVIDIA RTX 3090 Ti GPU. This configuration ensures a real-time decoding frame rate for 2k videos. According to common practice, The lowest buffer level, below which to download from the lowest bitrate, and the maximum buffer level allowed by the system are set to 10 and 25 seconds, respectively.

5 EVALUATION

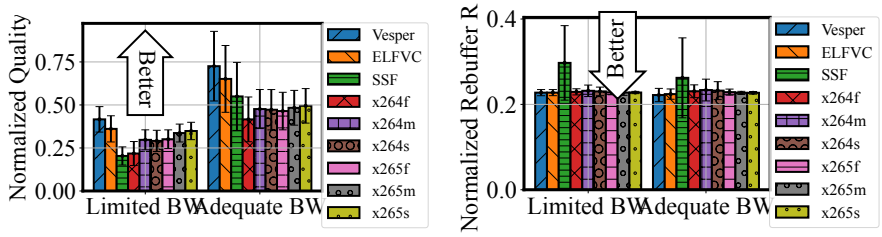
We evaluate Vesper regarding streaming performance and coding efficiency. Here are the key highlights of our findings:

(1) Superior QoE: Vesper demonstrates exceptional QoE compared to systems utilizing learned and traditional codecs, achieving improvements of up to 9% and 31% on average (Figure 7).
(2) Enhanced Rate-Distortion Trade-off: Vesper significantly improves the rate-distortion trade-off of both learned and traditional codecs (Figure 10).
(3) Super-precision Benefits: The application of super-precision leads to coding efficiency improvements, with up to a 0.4 dB increase in medium PSNR without additional bit consumption (Figure 13).
(4) Self Evolution Advantages: Self-evolution further enhances coding efficiency, resulting in up to a 0.8 dB improvement in medium PSNR and up to an 8% reduction in medium bits per pixel (bpp) (Figure 16).



(a) Average Normalized QoE (b) QoE CDF with limited bandwidth (c) QoE CDF with adequate bandwidth

Figure 7: End-to-end QoE. Vesper's QoE is up to 9% and 31% better than the best-performing learned and traditional approaches, respectively. Vesper improves the QoE of the second-best performing approach at medium by 9% and 8% at medium in the network with limited and sufficient bandwidth, respectively.



(a) Quality (PSNR) (b) Rebuffer Rate

Figure 8: Vesper has higher video quality (PSNR) than others at similar rebuffer rates.

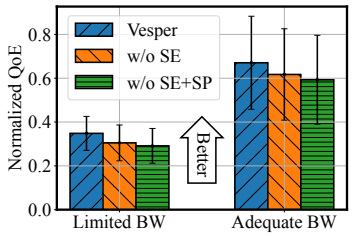


Figure 9: Ablation study of Vesper's QoE.

Baselines. In our evaluation, we consider state-of-the-art traditional codecs, namely x264 [54] and x265 [50], as well as learned video codecs, including SSF by Google [2] and ELFVC by OneWave [44], as baselines. For the traditional codecs, we utilize the FFmpeg [51] implementation and configure each codec with three modes: veryfast, medium, and veryslow. These modes are denoted as x264-veryfast (x264f), x264-medium (x264m), x264-veryslow (x264s), x265-veryfast (x265f), x265-medium (x265m), and x265-veryslow (x265s). Regarding the learned codecs, we employ the CompressAI [4] implementation for SSF. As ELFVC does not have an open-source implementation, we faithfully implement it based on the CompressAI framework. All these baselines are run on the same client machine as Vesper, with the traditional codecs primarily utilizing the CPU and the learned codecs mainly using the GPU. We compress videos with these learned codecs at eight compression levels, similar to Vesper, to ensure that their segments cover a sufficient range of bitrates.

Network traces. We utilize network traces obtained from the Federal Communications Commission (FCC) [16]. Randomly selecting 1,000 traces from two tests conducted for "video streaming" and "http get" scenarios, these traces represent a range of diverse network scenarios. The "video streaming" traces have an average bandwidth of 3.9 Mbps, indicating limited bandwidth availability, while the "http get" traces exhibit an average bandwidth of 15.8 Mbps, indicating more adequate bandwidth for streaming purposes.

Video datasets. In our study, we merge two video datasets, namely the UVG dataset [40] and the MCL-JCV dataset [53], to create a unified dataset. This unified dataset comprises 37 videos, each with a resolution of 2K and operating at a frame rate of 30 frames per second. The total runtime of this merged dataset is approximately 5 minutes.

Metrics. We compare the streaming performance of different approaches in terms of QoE (Equation 1), which is averaged over all videos in UVG and MCL-JCV and all network traces. In QoE, the video quality and rebuffering are evaluated by PSNR and rebuffer rate, respectively. Each metric is averaged over all videos in the dataset and all network traces, normalized with the maximum and minimum values mapped to 1 and 0, respectively. The error bar in our evaluation represents the standard deviation of the metric calculated over different network traces. While we assess the video quality by PSNR, a widely adopted quality metric, our system is generic to other video quality metrics, e.g., SSIM and VMAF, like ELFVC [44]. The only requirement is that the metric can be back-propagated. Moreover, we examine the coding efficiency by analyzing the trade-off between video quality (PSNR) and bitrate (bpp), which are averaged on all videos in UVG and MCL-JCV.

5.1 End-To-End Results

QoE. In Figure7(a), Vesper's average QoE is compared to other approaches. Vesper's normalized QoE surpasses the best-performing learned and traditional approaches by up to 9% and 31%, respectively. This result proves the effectiveness of Vesper and shows the advantage of learned video codecs in video streaming. The variance in QoE caused by bandwidth conditions results from varying emphasis on different sections of the rate-distortion curve, as shown later in Figure 10. As such, learned codecs have more advantages when the bitrate is higher, leading to a better QoE. Figures 7(b) and 7(c) demonstrate the QoE CDF of different approaches on two network traces. Vesper outperforms the second-best approach by

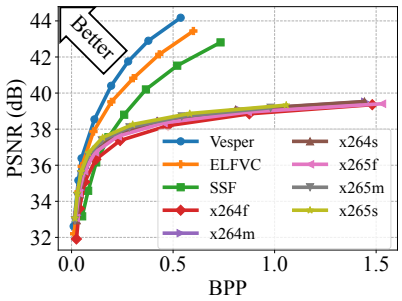


Figure 10: Vesper's coding efficiency compared to other codecs.

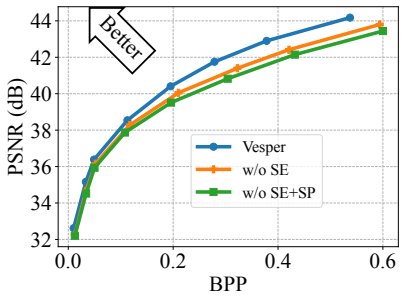


Figure 11: Ablation study of Vesper's coding efficiency .

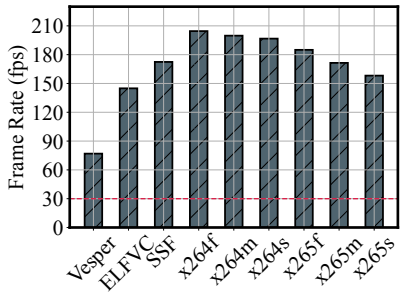
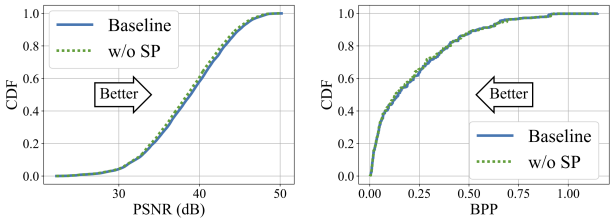
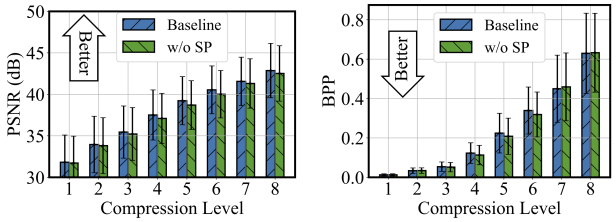


Figure 12: Vesper's frame rate compared to other codecs.



(a) PSNR (b) Bpp

Figure 13: CDF of PSNR and bpp with and without super-precision. Super-precision improves PSNR at medium by 0.4 dB with negligible impact on bpp.



(a) PSNR (b) Bpp

Figure 14: Impact of the compression level in super-precision. PSNR is increased by up to 0.5 dB on average. Bpp gets higher at some levels but decreases at others.

9% and 8% at medium in the network with limited and sufficient bandwidth, respectively. These results the advantage of Vesper is consistent under both limited and sufficient bandwidth.

QoE breakdown. In Figure 8, the QoE breakdown (quality and rebuffering) of various approaches is depicted. Vesper's QoE benefits most from the advantage in quality (PSNR). It achieves a superior average PSNR compared to other approaches, with improvements of up to 15% and 47% over the best-performing learned and traditional methods, respectively. In terms of the rebuffering rate, Vesper performs similarly to most other approaches. SSF shows slightly higher rebuffering rate than others.

Ablation. In Figure 9, a comparison is made between Vesper's QoE with different configurations: Vesper with both self-evolution and super-precision ("Vesper"), Vesper without self-evolution ("w/o SE"), and Vesper with only the base codec ("w/o SE+SP"). The results demonstrate the effectiveness of super-precision (§3.1) and self-evolution (§3.2) designs. The impact of self-evolution on QoE is more significant, which highlights the uncertainty in video content is better handled by Vesper than the uncertainty in quantization.

Frame rate. Figure 12 compares the frame rate of Vesper to other approaches. Though Vesper is not as fast as others mainly due to the super-precision module, it can attain a sufficient frame rate (77 fps) for streaming. In contrast to ELFVC, Vesper achieves nearly half the frame rate by adding single-pass denoising. It means that U-Net involves a substantial computation cost even if applied once, which must be applied with careful consideration of the computation constraints of the system.

Computation cost. The speed of self-evolution is 25 fps on average. While self-evolution takes 9 to 23 iterations of a video to complete, it does not stall video serving by running on the fly. As we show in Figure 15, self-evolution reflects on the viewer side immediately after one epoch, which achieves near-optimum performance for all videos. Considering the length of these videos being 300 to 600 frames, self-evolution only takes 12 to 24 seconds to reflect on the view side.

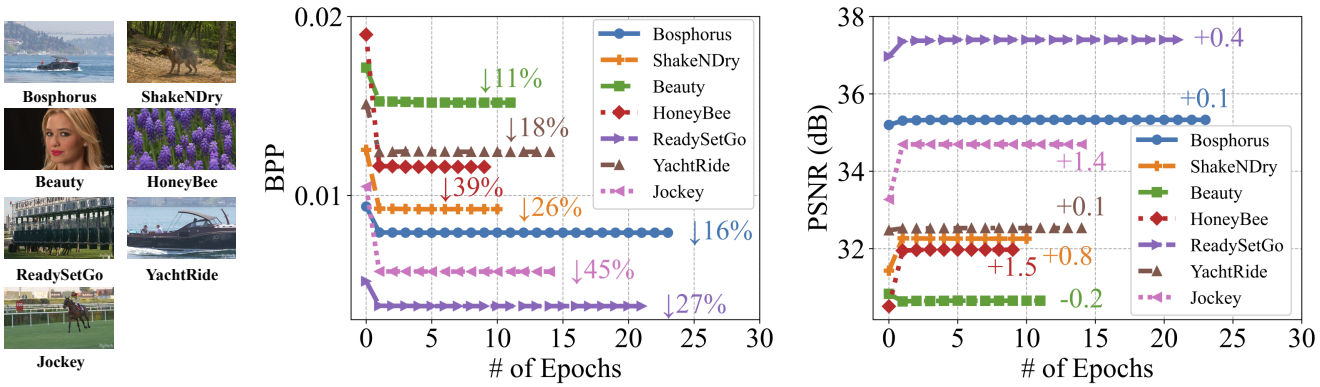
5.2 Coding Performance

Coding efficiency. Figure 10 illustrates the coding efficiency of Vesper, consistently surpassing systems driven by traditional and learned codecs. This advantage is more significant at higher bitrates. We speculate that Vesper can effectively exploit correlations when more information is encoded in feature values using a larger number of bits.

Ablation study. Figure 11 shows the coding efficiency of the default configuration ("Vesper") outperforms Vesper without self-evolution ("w/o SE"). Vesper without self-evolution also outperforms Vesper with only the base codec ("w/o SE+SP"). The results demonstrate that both self-evolution and super-precision contribute considerably to improving the coding efficiency of Vesper. Notably, self-evolution has a relatively more significant impact.

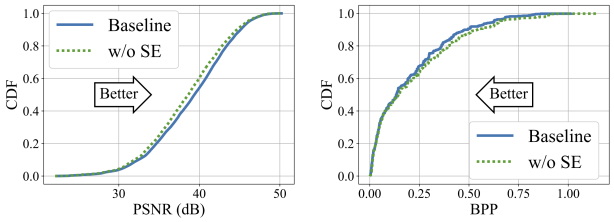
5.3 Super-precision

Analysis. In Figure 13, super-precision's effect on PSNR and bpp is depicted. The result shows that super-precision enhances PSNR by 0.4 dB at medium, indicating that super-precision better approximates original feature values than quantized features. We also notice the bpp metric remains largely unchanged. This is because super-precision is applied to quantized data, which minimally affects bit consumption. Moving on to Figure 14, it explores the impact of compression level on super-precision in terms of PSNR



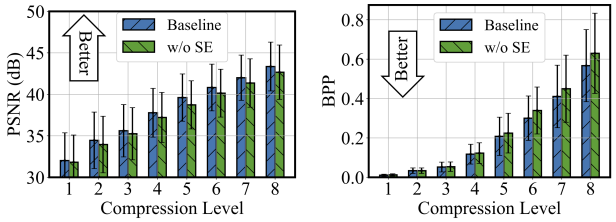
(a) Video Examples (b) Bpp (c) PSNR

Figure 15: Case study of self-evolution using seven videos from the UVG dataset.



(a) PSNR (b) Bpp

Figure 16: CDF of PSNR and bpp with and without self-evolution. Self evolution improves the medium PSNR by 0.8 dB while reducing the medium bpp by 8%.



(a) PSNR (b) Bpp

Figure 17: Impact of the compression level in self-evolution. Video quality (PSNR) and bit consumption are improved and saved by up to 0.7 dB and 14%, respectively.

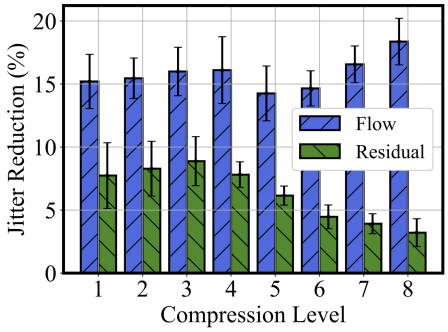


Figure 18: The reduction of feature jitter by super-precision in motion and residual auto-encoders.

and bpp. Higher compression levels correspond to higher video quality in terms of PSNR. Super-precision consistently enhances the average PSNR by up to 0.5 dB, with more substantial improvements observed at higher compression levels.

Jitter reduction. Figure 18 shows a reduction of up to 18% and 9% in feature jitter, measured by the L2 norm of motion and residual features at different compression levels. The motion and residual features are the output of motion and residual auto-encoders, respectively. The result demonstrates the effectiveness of super-precision in jitter reduction. Such reduction is more significant for the motion than the residual features. The reason is that motion features in a video are commonly spatially correlated, making it possible to leverage spatial correlation in the prediction of super-precision. Conversely, residual features, being the difference between raw and predicted frames, are more randomized and hard to predict.

5.4 Self-Evolution

Analysis. In Figure 16, the impact of self-evolution on PSNR and bpp is depicted. It demonstrates that self-evolution improves the medium PSNR by 0.8 dB while reducing the medium bpp by 8%. This indicates that self-evolution effectively enhances both video quality (PSNR) and bit consumption, resulting in a more efficient encoding process. Figure 17 explores the influence of compression level on self-evolution. It reveals a consistent improvement in video quality (PSNR) and reduction in bit consumption by up to 0.7 dB and 14%, respectively. Notably, the improvement in PSNR is more pronounced at higher compression levels. In terms of speed, self-evolution processes videos at a frame rate of 25 fps on an NVIDIA RTX 3090 Ti GPU.

Case study. Figure 15 illustrates the improvement achieved by Vesper in seven videos (Figure 15(a)) from the UVG [40] dataset. The compression level is set to 1 for clarity. Each point in Figure 15(b) and Figure 15(c) represents the attained bpp and PSNR, respectively, after a certain number of epochs, i.e., a complete pass of the video. The number of points indicates the number of epochs required for the optimization of a video to converge.

A consistent reduction of bpp by up to 45% (Figure 15(b)) and an improvement in PSNR by up to 1.5 dB (Figure 15(c)) are observed.

Although not all videos experience improved PSNR, the significant reduction in bpp ensures an overall enhancement in coding efficiency. Notably, the "HoneyBee" and "Jocky" videos demonstrate the highest reductions in bpp (39% and 45%) and improvements in PSNR (1.5 dB and 1.4 dB). This could be attributed to the presence of repetitive patterns such as purple-color flowers and green lawns, which are uncommon in regular videos but can be optimized through self-evolution to enhance coding efficiency. The reason for low PSNR improvement in certain content is the complexity of the content. For instance, the facial details and hair motions are challenging to exploit by Vesper in the "Beauty" video, with only a 0.2 dB reduction in PSNR. Nevertheless, self-evolution manages to reduce its bpp by 11%.

6 RELATED WORKS

Video codecs. Video codec standards, such as MPEG-2 [29], H.264 [54], and H.265 [50], have traditionally relied on handcrafted methods. However, recent advancements in deep learning have paved the way for DNNs to enhance the coding efficiency of these traditional codecs. Deep learning techniques have been employed to replace operations in traditional codecs [1, 3, 10, 11, 35, 43, 56] and serve as hints to assist video codecs [15, 33]. Moreover, there have been proposals for end-to-end learned codecs [2, 19, 31, 36, 37, 44, 59, 60]. Another category of learned approaches, exemplified by SWIFT [13], integrates DNN and progressive coding to enhance the QoE in streaming by leveraging available bandwidth to download more data. Among these learned video codecs, ELFVC [44] stands out as the state-of-the-art approach, greatly improving both speed and coding efficiency compared to prior works. However, existing video codecs fall short when handling the uncertainty of content and quantization, which undermines their coding efficiency. In this context, Vesper presents a novel approach by leveraging proactive learning to tackle the challenges posed by content and quantization uncertainty, achieving superior performance.

Streaming systems. Enhancing the quality of video streaming has been the focus of numerous systems, typically classified into three categories: push-based, pull-based, and video super-resolution (VSR). Push-based strategies analyze playback statistics from clients and deliver videos of suitable bitrates to each client from a central server. Various studies [7, 17, 23, 32] have investigated the effectiveness of these approaches. In contrast, pull-based strategies guide clients to download videos with appropriate bitrates from the server based on predicted bandwidth or buffer level [39, 48, 58, 62, 64, 65]. Additionally, VSR techniques can be employed to enhance video streaming quality by applying super-resolution models to increase the resolution of downloaded segments [25, 61, 63]. Our approach operates at the codec level of a streaming system, making it easily integrated into these distinct designs.

7 DISCUSSION AND FUTURE WORK

Integration with existing CDN infrastructure. While the existing CDN infrastructures might not exhibit the computation power to perform any model training in self-evolution, Vesper can still be seamlessly integrated with them. Specifically, the self-evolution can run on powerful video-processing servers and update video segments with optimized segments on CDN servers. Therefore, little computation is required on the CDN server. Moreover, on video-processing servers, as self-evolution is independent among different videos, Vesper can be further scaled via parallelization of the self-evolution of different videos. Although there will be an additional transmission delay between the video-processing server and the CDN server, such a delay would be amortized by the serving duration and the number of served users in on-demand streaming.

Compatibility with streaming protocols and ABR algorithms. In video streaming, Vesper upgrades the content of video segments with better coding efficiency. Since there is no change in the data format of the segments, Vesper is easily compatible with the streaming protocols and ABR algorithms that determine how to deliver these segments.

Extensibility to other learned codecs. The proposed self-evolution and super-precision modules do not make particular assumptions about the base codec as long as it is learning-based and involves quantization. Thus, these modules can be applied as general extensions to most existing learned codecs besides our base codec without a major change in principle.

Generalizability to different video contents. The design of Vesper is generic to the video content. While the specific content may affect the QoE gain from Vesper, our experiments demonstrate Vesper's consistent effectiveness.

Live video streaming. Vesper encounters challenges in live streaming, particularly due to the introduction of additional latency caused by self-evolution in the video encoding process. This latency is acceptable in on-demand streaming situations since it occurs on the fly, but this latency would become a critical concern in real-time live video streaming where encoding must happen without delay. As a result, the latency associated with self-evolution directly affects the end viewer, potentially leading to a negative impact on QoE.

Video heterogeneity. In this work, we assume every video is equally important and perform self-evolution to different video content in random order. However, videos may have different popularity in different applications. This factor could be utilized to perform self-evolution on more popular content first, which could further improve the QoE.

8 CONCLUSION

In this paper, we identify quantization uncertainty and content uncertainty that affect performance in video streaming. We present Vesper, a video streaming system that tackles uncertainty through learning-based modules of self-evolution and super-precision. Our experiments demonstrate the superior performance of Vesper in comparison to streaming systems built with state-of-the-art codecs, validating the effectiveness of handling uncertainty with learning.

ACKNOWLEDGMENTS

This work was supported by NSF under the award number NSF CNS 1900875, NSF CNS 2106592, NSF IIS 2140620, NSF IIS 2140645, NSF OAC 1835834, and NSF OAC 2144764 and Army Research Lab under the award number ARMY W911NF-17-2-0196.

REFERENCES

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. *arXiv preprint arXiv:1704.00648*, 2017.

[2] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020.

[3] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[4] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020.

[5] Fabrice Bellard. Bpg image format, 2018.

[6] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[7] Abdelhak Bentaleb, Ali C Begen, and Roger Zimmermann. Sdndash: Improving qoe of http adaptive streaming using software defined networking. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1296–1305, 2016.

[8] John-Michael Bond. How does hulu work? *Cord Cutters News*, October 2021.

[9] HBO Max Help Center. What devices and browsers are supported by hbo max? *HBO Max Help Center*, N/A 2021.

[10] Tong Chen, Haojie Liu, Qiu Shen, Tao Yue, Xun Cao, and Zhan Ma. Deepcoder: A deep neural network based video compression. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2017.

[11] Hyomin Choi and Ivan V Bajić. Deep frame prediction for video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(7):1843–1855, 2019.

[12] DASH Reference Client. Dash reference client, 2023.

[13] Mallesham Dasari, Kumara Kahatapitiya, Samir R Das, Aruna Balasubramanian, and Dimitris Samaras. Swift: Adaptive video streaming with layered neural codecs. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 103–118, 2022.

[14] Databox. Popular youtube video types. https://databox.com/popular-youtube-video-types, 2023. [Accessed: April 28, 2023].

[15] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. Server-driven video streaming for deep learning inference. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 557–570, 2020.

[16] Federal Communications Commission. Measuring broadband america - july 2012. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/measuring-broadband-america-july-2012, 2012. Accessed on April 21, 2023.

[17] Aditya Ganjam, Faisal Siddiqui, Jibin Zhan, Xi Liu, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. C3: Internet-scale control plane for video quality optimization. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 131–144, 2015.

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[19] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7033–7042, 2019.

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

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[22] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[23] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 393–406, 2017.

[24] Saravana Kannan and Anne Aaron. Per-title encode optimization. *Netflix Tech Blog*, December 2016.

[25] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 107–125, 2020.

[26] Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687. PMLR, 2018.

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

[28] Valérie Lasserre. Least absolute deviation estimation for arma models based on l1-norm innovations. *Signal Processing*, 89(7):1359–1369, 2009.

[29] Didier J Le Gall. The mpeg video compression algorithm. *Signal Processing: Image Communication*, 4(2):129–140, 1992.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[31] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. M-lvc: Multiple frames prediction for learned video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3546–3554, 2020.

[32] Xianshang Lin, Yunfei Ma, Junshao Zhang, Yao Cui, Jing Li, Shi Bai, Ziyue Zhang, Dennis Cai, Hongqiang Harry Liu, and Ming Zhang. Gso-simulcast: global stream orchestration in simulcast video conferencing systems. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 826–839, 2022.

[33] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[34] Renting Liu, Zhaorong Li, and Jiaya Jia. Image partial blur detection and classification. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[35] Zhenyu Liu, Xianyu Yu, Yuan Gao, Shaolin Chen, Xiangyang Ji, and Dongsheng Wang. Cu partition mode decision for hevc hardwired intra encoder using convolution neural network. *IEEE Transactions on Image Processing*, 25(11):5088–5103, 2016.

[36] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. Content adaptive and error propagation aware deep video compression. In *European Conference on Computer Vision*, pages 456–472. Springer, 2020.

[37] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

[38] Wei Luo and Bo Chen. Neural image compression with quantization rectifier. In *ICML 2023 Workshop Neural Compression: From Information Theory to Applications*, 2023.

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

[40] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 297–302, 2020.

[41] Peter Park and Ja-Yong Koo. Fast and stable signal recovery. *SIAM Journal on Scientific and Statistical Computing*, 11(2):273–294, 1990.

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[43] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4161–4170, 2017.

[44] Oren Rippel, Alexander G Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. Elf-vc: Efficient learned flexible-rate video coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14479–14488, 2021.

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[46] Gerard Salton and Michael J. McGill. Introduction to modern information retrieval. *Journal of the American Society for Information Science*, 37(3):173–174, 1986.

[47] Sandvine. Global internet phenomena report 2023. https://www.sandvine.com/global-internet-phenomena-report-2023, 2023. Accessed on May 3, 2023.

[48] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking*, 28(4):1698–1711, 2020.

[49] Bolan Su, Shijian Lu, and Chew Lim Tan. Blurred image region detection and classification. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1397–1400, 2011.

[50] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[51] Suramya Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[53] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1509–1513. IEEE, 2016.

[54] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[55] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[56] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 416–431, 2018.

[57] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.

[58] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Alexander Levis, and Keith Winstein. Learning in situ: a randomized experiment in video streaming. In *NSDI*, volume 20, pages 495–511, 2020.

[59] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6628–6637, 2020.

[60] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with recurrent auto-encoder and recurrent probability model. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):388–401, 2020.

[61] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. Neural adaptive content-aware internet video delivery. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 645–661, 2018.

[62] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 325–338, 2015.

[63] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. Yuzu:neural-enhanced volumetric video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 137–154, 2022.

[64] Rui-Xiao Zhang, Tianchi Huang, Ming Ma, Haitian Pang, Xin Yao, Chenglei Wu, and Lifeng Sun. Enhancing the crowdsourced live streaming: a deep reinforcement learning approach. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 55–60, 2019.

[65] Yuanxing Zhang, Pengyu Zhao, Kaigui Bian, Yunxin Liu, Lingyang Song, and Xiaoming Li. Drl360: 360-degree video streaming with deep reinforcement learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1252–1260. IEEE, 2019.