JitterNot: Adaptive Video Streaming in VCAs with DNN and MPC

Abstract

Video conferencing has become an essential part of everyday life worldwide. Since the COVID-19 pandemic, Video Conferencing Applications (VCAs) have become quintessential in the space of Education, Business, and Telehealth. How a VCA varies video quality under different network conditions is pivotal in determining the user-perceived quality of experience (QoE). Adaptive bitrate (ABR) algorithms are often used to optimize the quality of user experience (QoE) during video conferencing. In the client-side VCA, the predicted throughput and a video codec act together to improve the user’s QoE. They use fixed control rules or heuristics and so suffer from a key limitation: they don’t take into account the deployment environment and network conditions. We innovatively use the model-predictive controller (MPC) in a closed-loop and supply it with the predicted throughput to effectively evaluate QoE parameters. Our contributions are to use (i) an LSTM model to guide the adaptive bitrate algorithm, and then (ii) propose an ABR algorithm, JitterNot, that utilizes a closed-loop MPC controller, throughput prediction, and contextual information like Quantization Parameter (QP). We demonstrate that the closed-loop control by JitterNot can be an effective extension to the existing internal control by a codec in a VCA which might not neglect the tail of the network. We show that the video quality or QoE can improve or match the codec performance during a video conference.

1. Introduction

Due to the pandemic, our world was transformed and the use of Video Conferencing Applications (VCAs) has been encouraged worldwide across a majority of spheres affecting the lives of all including Education, the Corporate Sector, Medicine, and Telehealth. User-perceived quality-of-experience (QoE) is thus critical for the performance of a VCA and could have a profound impact on any of these fields. Issues such as stuttering, and low video quality at ample bandwidth can be detrimental and lead to a substandard user experience. Considering that there is little support in the network for optimizing QoE, bottlenecks could occur anywhere in the delivery system. Thus, a robust bitrate adaptation algorithm on the client-side is needed to ensure a good user experience. ABR for VCAs fundamentally differs from the ABRs used by leading video streaming services like Netflix [1] as these are built around the liberty of utilizing buffer occupancy.

The widely used video conferencing systems today include FaceTime, Hangouts, Skype, and WebRTC. These

applications are used for person-to-person videoconferencing, cloud video-gaming, teleoperation of robots and

vehicles, and any setting where video must be encoded and sent with low latency over the network. The standard approach used in the VCAs today involves two components: a transport protocol and a video codec [2]. During a video call, the transport protocol manages the transmission of the video stream packets to the receiver and processes acknowledgments, and congestion signals to estimate the average data rate of the network path. It supplies this estimate to the codec, a distinct module with its own internal control loop. The codec selects encoding parameters (a frame rate and quality setting like the Quantization Parameter) and generates a compressed video stream with an average bitrate that approximates the estimated network capacity.

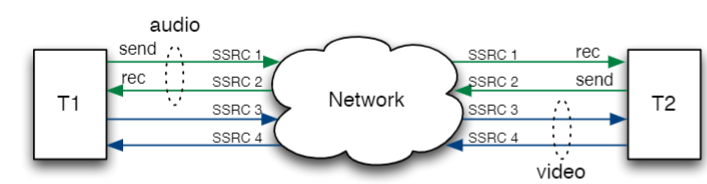
We use Google Meet as the VCA for all our experimentations and develop the closed-loop control for optimizing QoE, JitterNot, to be applied on top of the internal controls in place in Google Meet. The results in **section 4.3** indicate that JitterNot performs at least as well as the existing ABR mechanism of Google Meet outperforming it intermittently by our QoE definition. In general, we summarize our contributions as follows:

1. To the best of our knowledge, JitterNot is a one of its kind system to utilize deep learning (DL) techniques to tap the potential for MPC-based ABR algorithm for video transmission and is the first to do so for video conferencing.
2. We develop an effective throughput predictor using a long short-term memory network (LSTM) model to predict the throughput for future windows. Further, we also model QP, a valuable QoE metric using machine learning (ML) methods.
3. We curated a dataset for training our models using MahiMahi [3] to emulate different network conditions.

This paper proceeds as follows. Section ?? discusses background information on the real-time video system of Google Meet and related work including work on MPC. We describe the design and implementation of JiterNot in Section 3 where we describe the data collection process, the VCA as a control system, model it, and design the MPC controller. Section 5 presents the results of the evaluation. We discuss the limitations of the system and its evaluation in Section 6. JitterNot is publicly available, and the experiments reported in this paper are intended to be reproducible. The source code and raw data from the evaluation are available at https://github.com/SatyamA007/-JitterNot-Adaptive-Video-Streaming-in-VCAs-with-DNN-and-MPC

2. Related Work

WebRTC’s reference implementation has been incorporated into major Web browsers[[2]](https://www.usenix.org/system/files/conference/nsdi18/nsdi18-fouladi.pdf). We conducted our sessions on Google Meet (which uses WebRTC [4]) and used the Google Chrome browser. When using Google Chrome, data from both the sending and receiving parties in a WebRTC-based telemeeting can be gathered via the WebRTC internals page (chrome://webrtc-internals/). In our work, we examine two-party calls wherein one peer (sender A) will “share screen” to stream a video being played on a local video player. While the receiver (peer B) is muted and also has his camera turned off.

Figure 1: A two-party video conference call opens four tracks

**Performance Statistics in WebRTC-based Video Communication** [5] explain the peer connection between peers A and B. A unique SSRC ID (one per track) is shared between them. Peer A (sender), has stats labeled as sent (e.g.,bitsSentPerSecond) whereas peer B(receiver), has stats labeled as received (e.g., bitsReceivedPerSecond).

We will follow the convention of referring to a sender peer as A and the receiver as B in our report. For data collection **Section 3.1**, we vary the network conditions for A, keeping the conditions for B unrestricted.

**MDN Web Docs** [[6]](https://developer.mozilla.org/en-US/docs/Web/API/RTCRemoteOutboundRtpStreamStats/localId)gives us necessary details about the RTCStatsReport of a mediaStream captured using the WebRTC API: for A it includes a remote-outbound-rtp statistics object (of type RTCRemoteOutboundRtpStreamStats); it should also have a corresponding inbound-rtp object. Both of these provide information about the same batch of packets being transmitted from the remote peer to the local device. The difference is that remote-outbound-rtp describes statistics about the transmission(s) from the perspective of the remote peer (and so includes mostly empty or null values), while inbound-rtp offers statistics about the incoming data from the local peer's perspective. Thus, the inbound-rtp provides the feedback of video quality as perceived at the receiver B.

**2prong, an ABR solution with DNN and MPC** for video streaming [7] explored the design of ABR algorithms for Mobile network video streaming through Bayesian model predictive control-based approach 2prong. In contrast to state-of-art adaptation algorithms, 2prong takes Uncertainty-Aware Robust Adaptive video streaming transmission. It utilizes high-dimensional contextual information such as buffer occupancy, predicted throughput, and video quality to find the most valuable information for quality adaption in real-time. [7] demonstrated that 2prong can improve the video quality of video streaming transmission. Thus, it demonstrates that using a DNN for MPC controller for video streaming is relevant and motivates us to apply a similar approach for VCAs.

**ANN-based model predictive control (MPC) and optimization of HVAC systems** [8] is another work that uses neural networks based MPC but applies it to Heating, ventilation, and air conditioning (HVAC) systems. It states that traditionally, the industry has been very reluctant to adopt complicated methods in modeling and control of HVAC systems and due to this reason the majority of HVAC systems are still using very simple on/off or proportional-integral-derivative (PID) controllers instead of more modern controllers such as ANN-based model predictive control (MPC). We see similar skepticism in the adoption of any DNN or blackbox solutions in video conferencing. We take note of the important implementation specifics for the MPC technique and its controller architecture in [8].

**Deepmpc: A Mixture ABR Approach via DL and MPC** [9]shows that the fusion of DL and MPC is not only

more effective and interpretable but also achieves state-of-the-art performance compared with existing algorithms. DeepMPC enhances MPC via two deep learning-based modules, i.e., DL-based Throughput Predictor (DTP), which can precisely predict future bandwidth, and Discounted Factor Optimizer (DFO), which estimates the prediction error. Moreover, its implementation in real-world network environments and experimental results demonstrate the superiority of DeepMPC against existing state-of-the-art approaches. Like [7], [9]is also an ABR solution for video streaming and utilizes buffer occupancy.

3. JitterNot Design

3.1. Collecting data

In order to develop a data-driven model such as RNN or LSTM, a rich dataset is required that captures most of the network conditions. However, tackling a novel problem like ours (designing an ABR framework for VCAs) has the downside of no existing dataset that is publicly available. Thus, the dataset had to be curated independently with a provision for fine adjustment of the network link parameters to emulate different network conditions. Also, the neural network(s) we develop should be relatively uncomplicated for the model to fit the training data well.

MahiMahi’s [3] network emulation tools can be used to emulate many different link conditions. Mahimahi supports emulating fixed propagation delays (DelayShell), fixed and variable link rates (LinkShell), and stochastic packet loss (LossShell). LinkShell also supports various queueing disciplines such as DropTail, DropHead, and active queue management schemes like CoDel. Each of Mahimahi's network emulation tools can be arbitrarily nested within one another, providing more experimental flexibility. Hence, MahiMahi was chosen as the tool for emulating various network conditions.

The webrtc-internals functionality for Google Chrome has been used by WebRTC application developers to understand the features and functions of their WebRTC services. It is relatively new to explore and utilize these stats to study the QoE aspects of WebRTC services. It enables observation of the performance of the WebRTC connections locally in the browser. During a 10-day test period, we set up video conference sessions on Google Meet, a WebRTC application, in the Google Chrome browser running in the link, delay, and loss shells - running as nested instances to provision varying network conditions. A mix of network conditions was applied - latency from 0 to 1500ms, link packet loss from 0 to 35%, and link speed from 3 Mbps to 100mbps. Each video session was kept up for 3-4 minutes giving the total volume of the video conference statistics dataset amounting to 6-7 hours.

The sampling resolution of the webRTC internals tool is 1 second. Thereby, we got over 25,000 data points each containing features giving various performance details of the video conference. Also, for each session, the same video was played in a video player offline in the test machines. This helped us to produce accurate results for two reasons: (i) the network in the test machine had only the video conference traffic and not the traffic from streaming the video online, and (ii) video segments with high movement scenes are likely to be encoded with a high bitrate before transmission, effectively having a variable bitrate (VBR). Having the same video played throughout the training data would ensure a good fit in our neural network models as each session will have a similar VBR. Although this might not be ideal to generalize for other videos, we expect our MPC controller to make precise decisions for high QoE as it is supplied with feedback. Also, this could be addressed by adding to the data set and using more real-world scenarios during a video conference.

**3.2. System Modeling**

In this section, we present an outline of JitterNot (Figure 2) that helps describe the ongoing video conference process, running our QoE optimization solution, as a closed-loop controlled system in the time (t) domain. Also, we emphasize explaining in detail the steps for modeling the system and its components. Section ?? will cover the specifics for the controller.

Figure 2 highlights 4 primary components of the system:

1. WebRTC Stats: The Google Meet VCA, uses webRTC, an open framework for the web that enables Real-Time Communications in the browser. The webRTC API allows capturing the mediaStream statistics including all the features we need.
2. Throughput Prediction: Uses an LSTM model to predict the throughput values (in packets/s) for the horizon size of 5 seconds
3. MPC Controller: Includes 2 models for QoE estimate (i) frame jitter, which applies a formula to incoming data to obtain the estimate of the frame jitter component of the QoE, and (ii) QP, a polynomial regression model of the quantization parameter, another component of our QoE. It solves the QoE optimization problem using a constraint solver.
4. Video Resolution Switching: It receives the resolution values that optimize the QoE. And changes the resolution of the ongoing video call.

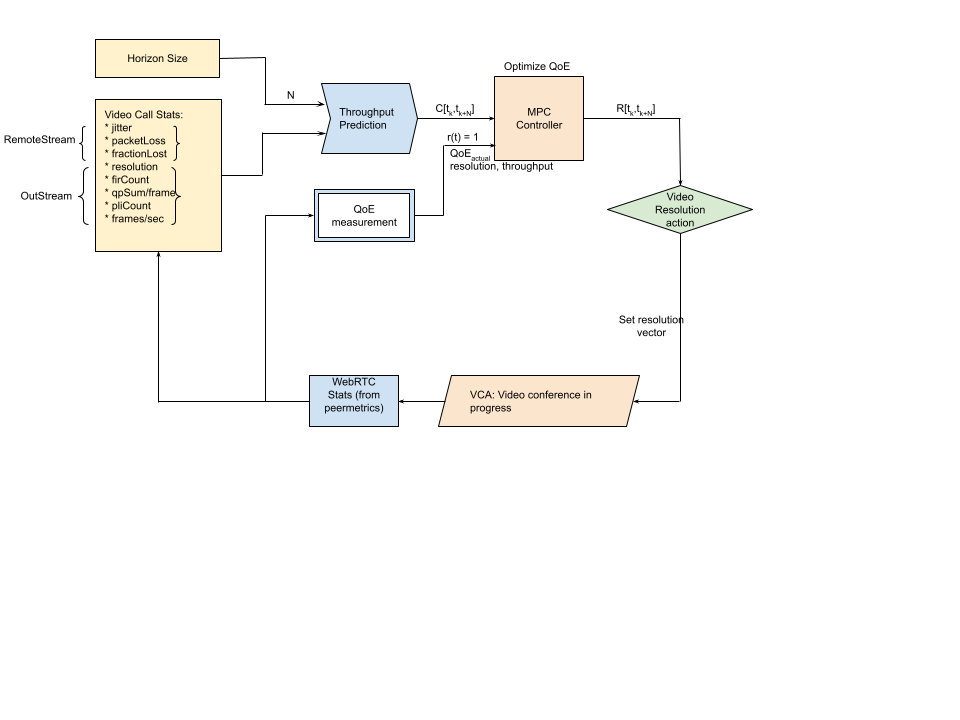


Figure 2: Block Diagram for JitterNot

**3.2.1. Feature Selection\*\*[Nikunj]**

Explain why some features might be better than others

**3.2.2. LSTM model for throughput prediction\*\* *[Nikunj]***

Clarify training techniques, x/y etc

**3.3. QoE Metric**

There exists significant variance in user preferences for video streaming QoE. We consider a variety of QoE factors. We use the general QoE metric used by MPC [10] and modify it for relevance in VCA application, it is given as

and normalized

where α, β, γ are constants that normalize the QoE terms q(R), P and F

for a video conference at time k: Rk is the resolution of the video, q(Rk) maps that resolution to the quality perceived by a user, Pk is qpSum/sec or QP at k. Fk is the frame jitter term, is the average delay variation for downloading a frame in the past 5 seconds i.e. for time [k-4, k].

We consider the choice of q(Rn ) = log(R/Rmin). This metric captures the notion that, for some users, the marginal improvement in perceived quality decreases at higher bitrates. The avg and min values for any term is calculated from the past 10 second window. We define the individual terms in our QoE metric in detail.

**3.3.1. Resolution**

Adobe defines video resolution as the number of pixels contained in each frame [11]. Video resolution determines the amount of detail in your video, or how realistic and clear the video appears. It’s measured by the number of pixels contained in a frame. A higher number of pixels indicates a higher resolution, and a lower number of pixels makes for a low-resolution video.

We define resolution Rk to be a product of frame width and frame height at time k. It captures the amount of data or pixels in each frame so it is proportional to the frame size. Moreover, we will use Rk as the manipulated variable for our MPC controller.

**3.3.2. Frame Jitter Estimation**

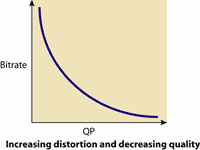
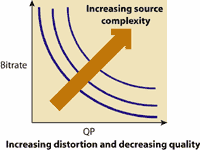
Jitter is the amount of variation in packet delay, which is why it is also frequently called delay variation. It is measured by calculating the average variation in packet arrival times at the receiving node occurring over a fixed interval. At callstats.io, the jitter calculation interval varies between five and ten seconds depending on endpoint configuration. [[12]](https://www.callstats.io/blog/why-is-jitter-an-important-webrtc-metric-for-contact-centers)

Frame jitter can thus be derived from network jitter, by modifying the definition to measure delay variations for each frame instead of a single packet and so it varies with the resolution (or total pixels) of the frame Rk., where a is model parameter, C bandwidth or throughput

**3.3.3. Polynomial regression model for Quantization Parameter**

Pixeltools describes how any codec dynamically adjusts encoder parameters to achieve a target bitrate. Which is relevant for the video codec in case of VCAs as well. Block-based hybrid video encoding schemes such as the MPEG and h.26\* families are inherently lossy processes, this includes the VP9 codec used in Google Meet [13].

In particular, the quantization parameter QP regulates how much spatial detail is saved. When QP is very small, almost all that detail is retained. As QP is increased, some of that detail is aggregated so that the bit rate drops – but at the price of increase in distortion and loss of quality. Figure 3.1 suggests that relationship for a particular input picture – if you want to lower bit rate, you can do so by lowering QP at a cost of increased distortion. Figure 3.2 suggests that as source complexity varies during a sequence, you move from one such curve to another.

Figure 3: (1, left)For a particular source frame, (2, right) But when source complexity varies

Thus, we try to model the QP (qpSum/s) or P(t) from the bandwidth C and resolution R as it represents the frame size, and their ratio proportional to the bitrate. And use polynomial regression for the same.

where x represents C(t)/R(t) and coefficients are model parameters

**3.4. Controller Architecture**

The key idea of our MPC is to maximize QoE (expressed in terms of video quality terms) of users via a model predictive control method during the entire session. For each N second horizon, MPC first uses a DL-based throughput predictor to estimate future bandwidth [Ck,,Ck+N]for the horizon. It then simplifies the frame jitter Fk and qpSum/s Pk from the predicted Ck. These are expressed in terms of unknown resolution Rk, which is obtained by maximizing the QoE, a normalized combination of these parameters, subject to dynamic constraints.

**3.4.1. Why MPC?**

First of all, model predictive control may be naturally suitable for the problem of bitrate adaptation [[7]](https://ieeexplore.ieee.org/document/9751556). Though we cannot say that MPC is the best choice among all possible ABR algorithms. We can say that the MPC algorithm is more effective in mobile network video conferencing. Ideally, for a video call [tk, tk+1], the QoE optimization problem can be directly calculated by the optimal resolution R1,... , Rk and jitter, and qp.

However, in practice, we do not have access to this perfect information, making it difficult to optimize the solution offline. Despite the fact that perfect information for the future as a whole is not available, it is possible to obtain reasonably accurate throughput predictions over a short span of [tk, tk+N ] in the near future.

Owing to the network conditions (and even video call metrics) being quite stable over a short period of time and not changing drastically within a few tens of seconds. We can use this feature of throughput to run QoE optimization by applying the first resolution Rk and moving the horizon forward to [tk, tk+1]. This scheme is known as model predictive control (MPC). The advantage of MPC is that MPC can use prediction to optimize complex control objectives in dynamic systems online, subject to constraints.

**3.4.2 MPC Controller**

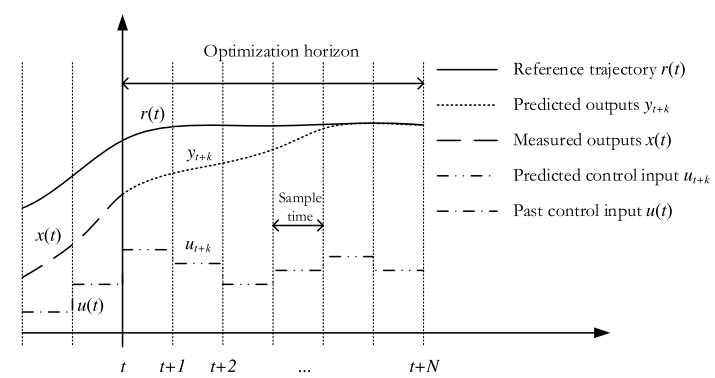


Figure 4: Inputs and outputs of MPC-based controller

Figure 4 shows the inputs and outputs of the MPC controller [[8]](https://www.researchgate.net/publication/313623952_Artificial_Neural_Network_ANN_based_Model_Predictive_Control_MPC_and_Optimization_of_HVAC_Systems_A_State_of_the_Art_Review_and_Case_Study_of_a_Residential_HVAC_System). MPC uses a system model to predict the future states of the system and generates a control vector (manipulated variable MV) that minimizes a certain cost function over the prediction horizon in the presence of constraints. The first element of the computed control vector at any sampling instant is applied to the system input, and the remainder is discarded. The entire process is repeated the next instant. Cost function can take the form of tracking error, control effort, frame jitter, low-resolution penalty, qpSum, or a combination of these factors.

Constraints can be placed on the rate and range limits of manipulated variables (e.g., upper and lower limits of jitter, qpSum, and range limits for resolution). This effort results in a controller that is robust to both time-varying disturbances and system parameters and regulates the process tightly within given bounds.

MPC performs the following three key steps: predict, optimize, and apply.

**1) Predict:** We use Deep RNN to predict throughput, simplify the jitter Fk and QP Pk subsystems, and use respective CNN models for current values that only depend on the manipulated variable, resolution Rk.

**2) Optimize:** This is the core of the MPC algorithm: Given the previous resolution R[tk-M,tk-1], jitter F[tk-M,tk-1], QP P[tk-M,tk-1], and throughput prediction C[tk,tk+N], find optimal resolution Rk.   
Rk = fmpc(R[tk-M,tk-1], P[tk-M,tk-1], F[tk-M,tk-1], C[tk,tk+N]), implemented by solving QoE\_MAX. fmpc also includes current jitter Fk and framerate Pk terms which are expressed using Rk as the input to their respective models.

**3) Apply:** Use the R[tk,tk+N] control vector as resolution in the VCA for optimized QoE in the horizon [tk,tk+N].

**3.4.3. Problem Formulation**

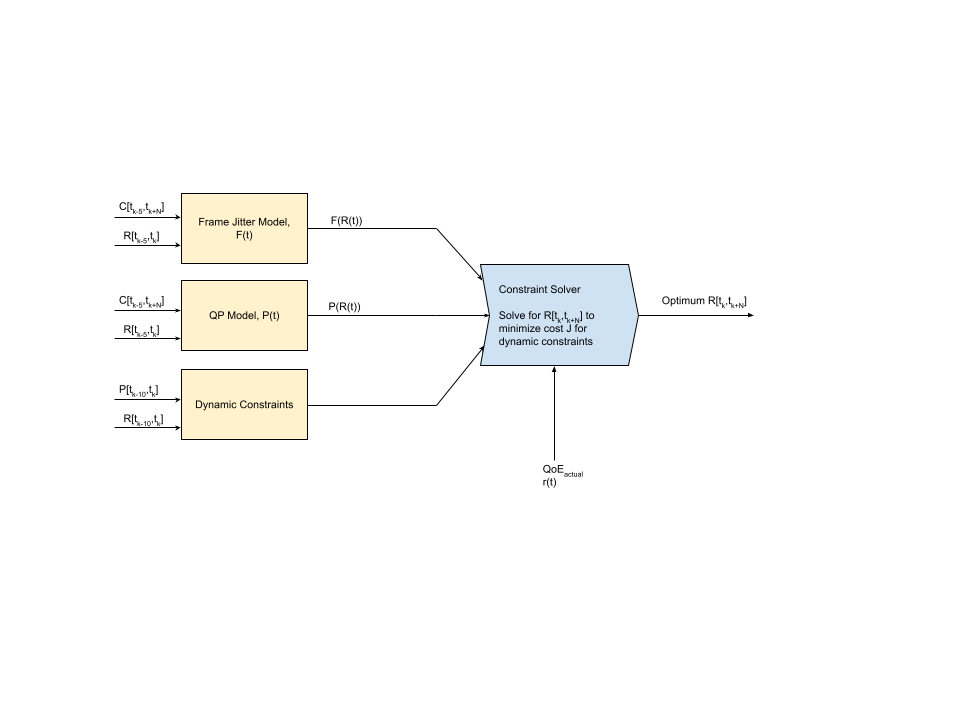


Figure 5: MPC Controller architecture

In this section, we formally describe the optimization problem, give details of the constraints and cost function considered by our MPC controller, and how we embody the ABR algorithm into the MPC framework.

Consider a multi-input single-output (MISO) plant S, with input u and output y signals sampled at a regular time interval Ts [[14]](https://ieeexplore.ieee.org/document/8698829). We aim at synthesizing a controller C for S such that the controlled system achieves a

desired engineering objective defined in terms of minimization of a cost *J(y:T , {u1:T} )*, where *u1:T* denotes the sequence of input signals measured at time steps t = 1,..., T, and T is the length (measured in the number of samples) of the experiment where the performance is measured.

The inputs u1 to S in our problem will beframe jitter F, resolution R, and QP P. However, as mentioned in section ??, since we model F and P and can express them in terms of R, our MISO system gets simplified to a SISO or single-input single-output system. The output y represents QoE.

Besides minimizing the cost *J(y:T , {u1:T} )*, the following constraints on inputs and outputs should be satisfied:

*umin ≤ u(t) ≤ umax* (1)

which for our problem is given as:

*0 ≤ F(t) ≤ 1.5Fmax,* (1a)

*Rmin/2≤ R(t) ≤ Rmax* (1b)

*0≤ P(t) ≤ 1.5Pmax* (1c)

Constraints (1) impose the domain limitation for the inputs and, gives little room for the performance metric to worsen since the optimization solution may exist wherein one or more inputs are worse but provides the optimal QoE under given conditions. The control design problem is formulated as the following optimization problem:

*min J(y:T , u1:T ) s.t. (1) ,* (2)

*RkR…………………………………*…………….……………………………………..

where the cost *J(y:T , {u1:T} )* is the summation of mean squared error (MSE) over the horizon for r(t) = 1 as the reference trajectory, since value 1 for the normalized QoE, , is desired.

*J(y:T , u1:T ) =*

**4. Evaluations**

**4.1 Throughput prediction using LSTM[Nikunj]**

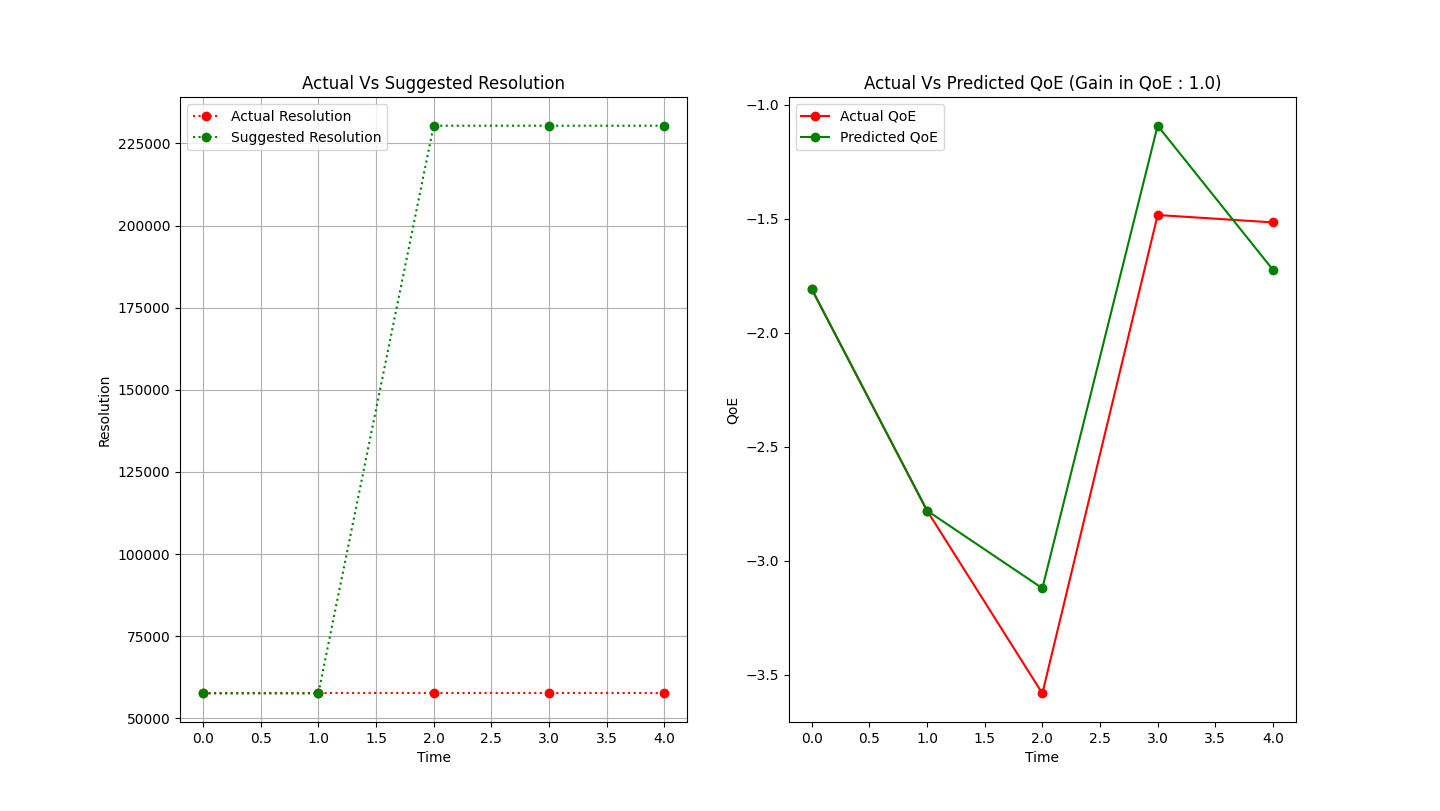
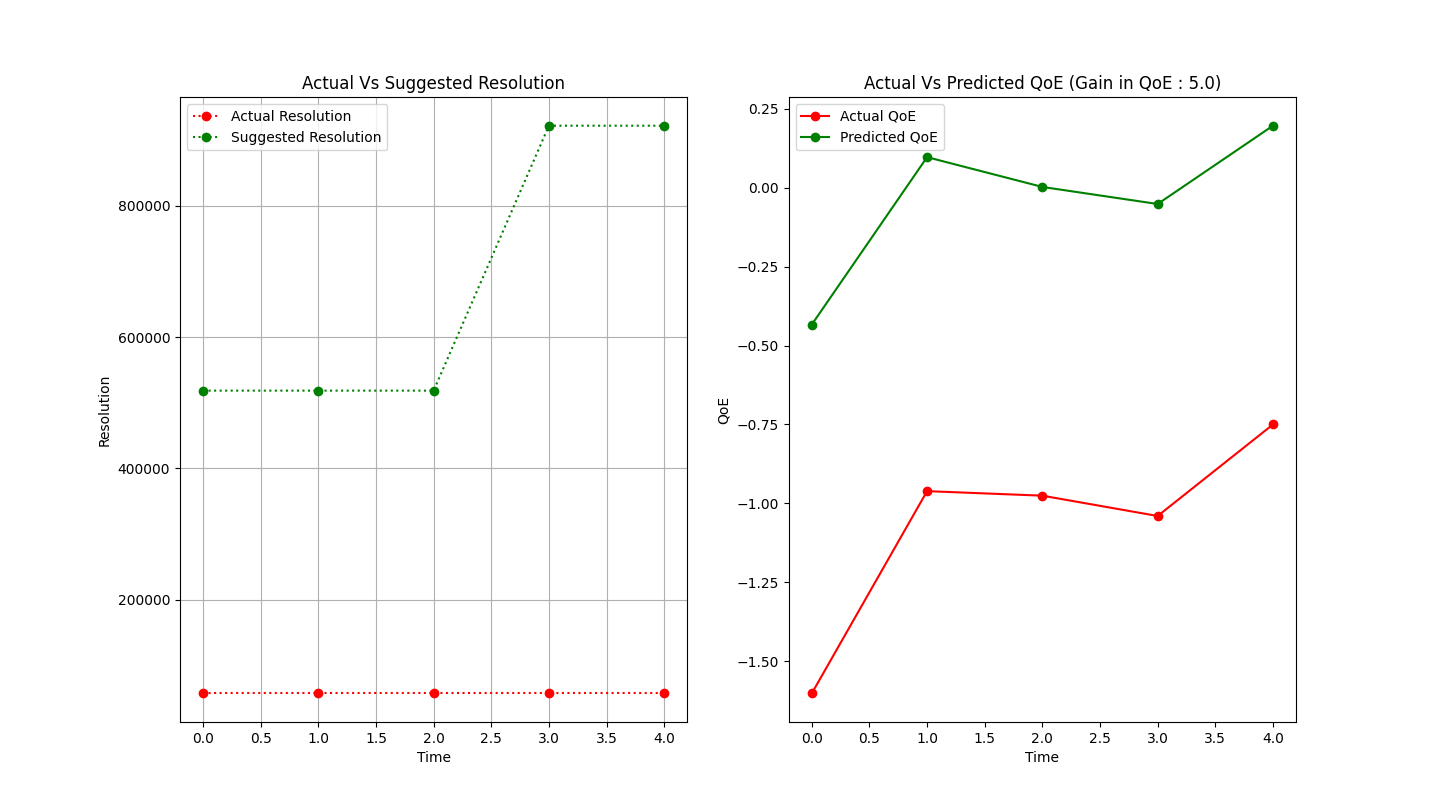
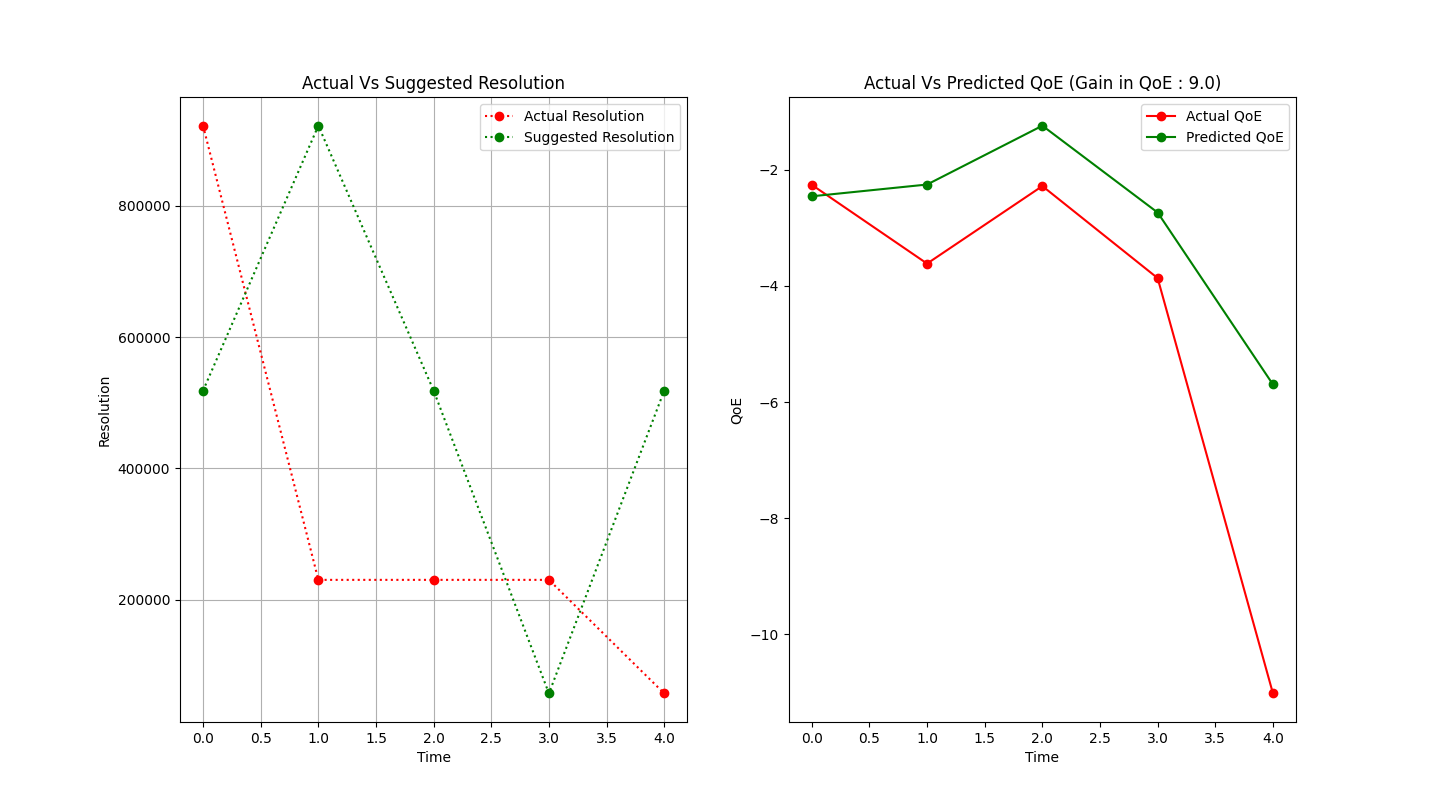
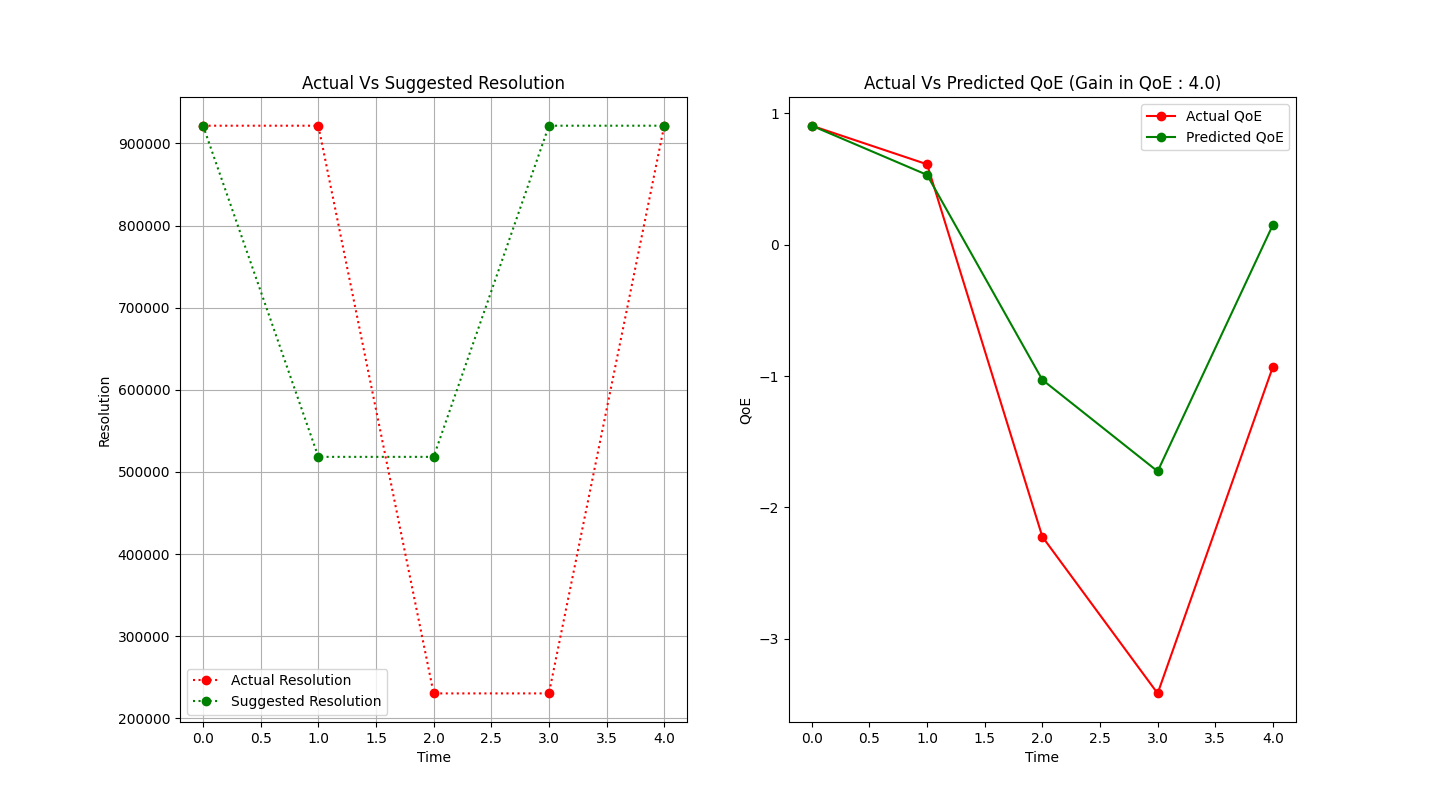
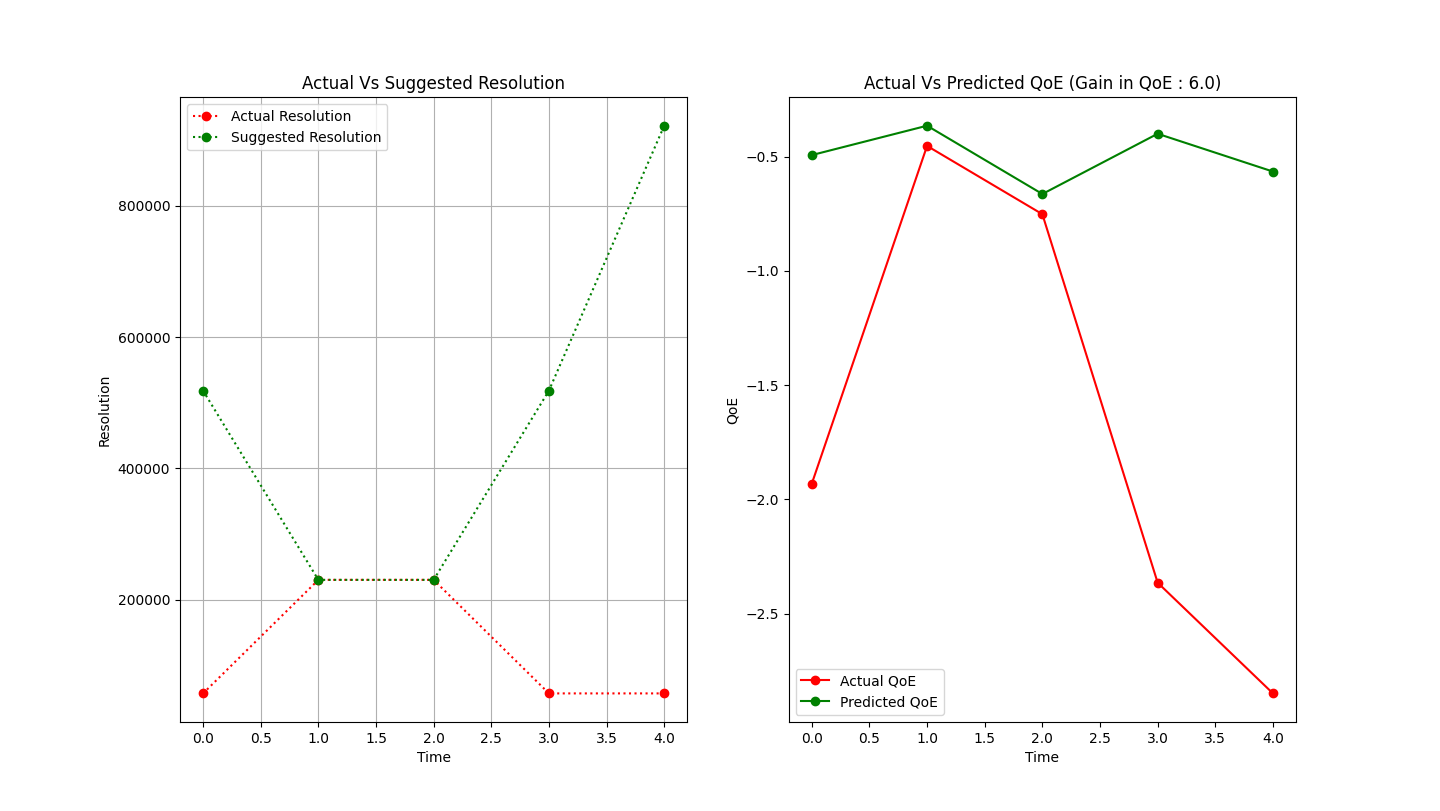
What is x,y, explain results, compare AutoML, why we chose LSTM; sMAPE score

**4.2 QP using Polynomial Regression [Nikunj]**

Like above

**4.3 JitterNot vs GoogleMeet’s existing ABR**

Below figure (Figure 6) provides a comparison between JitterNot’s suggested resolution as compared to Google Meet’s actual resolution. We have captured and analyzed JitterNot’s MPC performance across 10 different video segments from our dataset where each segment spans 5 seconds. We also compare the gain in QoE that JitterNot achieves against using Google Meet. **Section 3.3** explains the QoE metric evaluation and the details for frame jitter and quantization parameter is explained in the following sections. Almost every case shows an improvement in the Resolution when compared to Google Meet’s actual resolution.



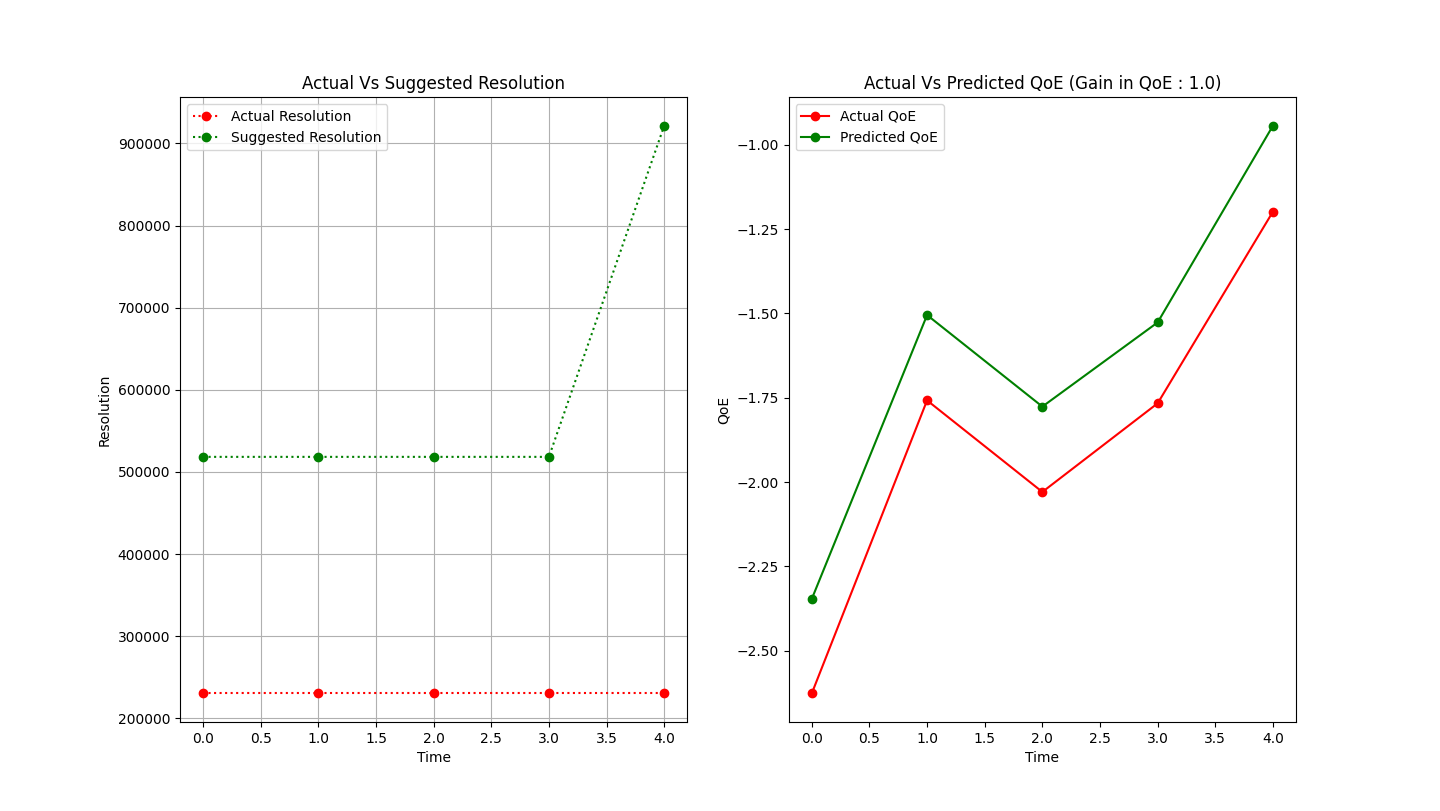


Figure 6: MPC Controller architecture

**5. Conclusion**

In this work we presented JitterNot, an innovative ABR approach for real-time video conferencing with the fusion of DL and model-predictive control methods. JitterNot improves upon the existing ABR approaches for VCAs as they use fixed control rules or heuristics so, suffer from the limitation of not taking the deployment environment and network conditions in consideration. The salient features of our closed loop control system are: (i) an LSTM model to guide the adaptive bitrate algorithm, (ii) an ABR algorithm, that utilizes an MPC controller, throughput prediction, and contextual information like Quantization Parameter (QP) and frame jitter.

In our evaluations, we compared our approach to the existing one in Google Meet and found that JittterNot often achieved better QoE scores, a metric composed of contextual information like resolution, QP, and frame jitter. And, so had a better video quality, lower delay variation for transmitting frames, and compression level denoted by QP. However, since our solution acted on top of another internal control loop inside the VCA’s video codec, there might have been an interplay between them. This dynamic interaction between the loops, if strong enough, could cause problems like longer response times and higher jitter, though it was not apparent during evaluations. Since disabling the internal controller would solve the issue of loop interaction, we are optimistic that our solution might outperform in the case of open-source VCAs that allow this. Another concern is the decision-making time that depends on the constraint solver since timely action is crucial for QoE. On our setup, it took 100-400ms for every decision which is high for a client-side solution. This could be addressed by delegating the solving tasks to an external server.

The results suggest that improvements can be made to the current techniques used by commercial VCAs. And that there is potential for a DL-based solution since, unlike rule-based approaches, they can account for network conditions in different deployment environments and adapt to them.

**References**

[1] Hongzi Mao, Ravi Netravali, Mohammad Alizadeh; Neural Adaptive Video Streaming with Pensieve; Proceedings of the 2017 ACM SIGCOMM Conference; ACM: [pdf](https://web.mit.edu/pensieve/content/pensieve-sigcomm17.pdf)

[2] Sadjad Fouladi, John Emmons, and Emre Orbay, Catherine Wu, Riad S. Wahby and Keith Winstein; Salsify: Low-Latency Network Video; 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’18). April 9–11, 2018; NSDI: [fouladi](https://www.usenix.org/conference/nsdi18/presentation/fouladi)

[3] Netravali, Ravi, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. "Mahimahi: Accurate {Record-and-Replay} for {HTTP}." In 2015 USENIX Annual Technical Conference (USENIX ATC 15), pp. 417-429. 2015; [pdf](http://mahimahi.mit.edu/mahimahi_atc.pdf)

[4] Calvin Nguyen, WebRTC — The technology that powers Google Meet, Medium (online publishing platform) In Mar 6, 2020; [available](https://medium.com/swlh/webrtc-the-technology-that-powers-google-meet-hangout-facebook-messenger-and-discord-cb926973d786)

[5] D. Ammar, K. De Moor, M. Xie, M. Fiedler and P. Heegaard, "Video QoE killer and performance statistics in WebRTC-based video communication," 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), 2016, pp. 429-436, doi: 10.1109/CCE.2016.7562675. [pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7562675)

[6] RTCRemoteOutboundRtpStreamStats, MDN Web Docs (Resources for Developers), Modified In Jun 1, 2022; [available](https://developer.mozilla.org/en-US/docs/Web/API/RTCRemoteOutboundRtpStreamStats/localId)

[7] Y. Wang, T. Zhou, C. Hou, B. Hou and Z. Cai, "2prong: Adaptive Video Streaming with DNN and MPC," 2021 17th International Conference on Mobility, Sensing and Networking (MSN), 2021, pp. 726-731, doi: 10.1109/MSN53354.2021.00114. msn: [pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9751556)

[8] Abdul Afram, Farrokh Janabi-Sharifi, Alan S. Fung, Kaamran Raahemifar, Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems; Volume 141, 2017, Pages 96-113, ISSN 0378-7788; elsevier: [pdf](https://www.sciencedirect.com/science/article/pii/S0378778816310799/pdfft?isDTMRedir=true&download=true)

[9] T. Huang and L. Sun, "Deepmpc: A Mixture Abr Approach Via Deep Learning And Mpc," 2020 IEEE International Conference on Image Processing (ICIP), 2020, pp. 1231-1235, doi: 10.1109/ICIP40778.2020.9191198; icip: [pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9191198)

[10] Yin, Xiaoqi, 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, pp. 325-338. 2015; acm: [pdf](https://conferences.sigcomm.org/sigcomm/2015/pdf/papers/p325.pdf)

[11] A beginner’s guide to video resolution, Adobe (Computer software company); [available](https://www.adobe.com/creativecloud/video/discover/video-resolution.html)

[12] Why is Jitter an Important WebRTC Metric for Contact Centers?, CallStats.io (The WebRTC call quality tool); In January 21, 2019; [available](https://www.callstats.io/blog/why-is-jitter-an-important-webrtc-metric-for-contact-centers)

[13] Dynamically adjust encoder parameters to achieve a target bitrate, PixelTools(Video Captioning And Processing Experts); [available](https://www.pixeltools.com/rate_control_paper.html)

[14] D. Piga, M. Forgione, S. Formentin and A. Bemporad, "Performance-Oriented Model Learning for Data-Driven MPC Design," in IEEE Control Systems Letters, vol. 3, no. 3, pp. 577-582, July 2019, doi: 10.1109/LCSYS.2019.2913347; [pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8698829)

[15] JitterNot-Adaptive-Video-Streaming-in-VCAs-with-DNN-and-MPC, GitHub Repository for JitterNot; [available](https://github.com/SatyamA007/-JitterNot-Adaptive-Video-Streaming-in-VCAs-with-DNN-and-MPC)